```
ASM80 PUTF.ASM DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1
PUTF - CP/M 3.0 Output Redirection - August 1982

  LOC  OBJ         SEQ         SOURCE STATEMENT

                     1 $title  ('PUTF - CP/M 3.0 Output Redirection - August 1982')
                     2 ;*****************************************************************
                     3 ;
                     4 ;       PUT  'Redirection Initializer'  version 3.0
                     5 ;
                     6 ;       11/30/82 - Doug Huskey
                     7 ;*****************************************************************
                     8 ;
                     9 ;
                    10 ;       Copyright (c) 1982
                    11 ;       Digital Research
                    12 ;       P.O. Box 579
                    13 ;       Pacific Grove, Ca.
                    14 ;       93950
                    15 ;
                    16 ;
                    17 ;       generation procedure
                    18 ;
                    19 ;       seteof put.plm
                    20 ;       seteof getscan.dcl
                    21 ;       seteof putf.asm
                    22 ;       seteof getscan.plm
                    23 ;       seteof parse.asm
                    24 ;       is14
                    25 ;       asm80 putf.asm debug
                    26 ;       asm80 mcd80a.asm debug
                    27 ;       asm80 parse.asm debug
                    28 ;       plm80 put.plm pagewidth(100) debug optimize
                    29 ;       link mcd80a.obj,put.obj,parse.obj,putf.obj,plm80.lib to put.mod
                    30 ;       locate put.mod code(0100H) stacksize(100)
                    31 ;       era put.mod
                    32 ;       cpm
                    33 ;       objcpm put
                    34 ;       rmac putrsx
                    35 ;       link putrsx[op]
                    36 ;       era put.rsx
                    37 ;       ren put.rsx=putrsx.prl
                    38 ;       gencom put.com
                    39 ;       gencom put.com put.rsx
                    40 ;
                    41 ;
                    42 ;       This module is called as an external routine by the
                    43 ;       PL/M program PUT.  The address of a the following
                    44 ;       structure is passed:
                    45 ;
                    46 ;                 declare putpb structure
                    47 ;                   (output$type   byte,
                    48 ;                    echo$flag     byte,
                    49 ;                    filtered$flag byte,
                    50 ;                    system$flag   byte);
                    51 ;
                    52 ;       output$type     = 0     > console output (default)
```

CP/M - PLUS V. 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  53 ;                    = 1     > auxiliary output
                  54 ;                    = 2     > list output
                  55 ;                    = 3     > console input
                  56 ;
                  57 ;       echo         = true  > echo output to real device
                  58 ;                               (default)
                  59 ;                    = false > don't echo output (input is
                  60 ;                               still echoed)
                  61 ;       filtered     = true  > convert control characters
                  62 ;                               to a printable form
                  63 ;                               preceeded by an ^
                  64 ;                    = false > no character conversions
                  65 ;       program      = true  > continue until user uses
                  66 ;                               PUT command to revert to
                  67 ;                               console
                  68 ;                    = false > active only until program
                  69 ;                               termination
                  70         public  putf
                  71         extrn   mon1,fcb,memsiz
                  72 ;
                  73 ;
FFFF              74 true           equ     0ffffh
0000              75 false          equ     00000h
                  76 ;
FFFF              77 biosfunctions  equ     true          ;intercept BIOS list or conout
                  78 ;
                  79 ;
                  80 ;       low memory locations
                  81 ;
0000              82 wboot    equ     0000h
0001              83 wboota   equ     wboot+1
                  84 ;
                  85 ;       equates for non graphic characters
                  86 ;
000D              87 cr       equ     0dh     ; carriage return
000A              88 lf       equ     0ah     ; line feed
                  89 ;
                  90 ;       BDOS function equates
                  91 ;
0001              92 cinf     equ     1       ;read character
0002              93 coutf    equ     2       ;output character
0006              94 crawf    equ     6       ;raw console I/O
000A              95 creadf   equ     10      ;read buffer
000B              96 cstatf   equ     11      ;status
0005              97 lchrf    equ     5       ;list character
0009              98 pbuff    equ     9       ;print buffer
000D              99 resetf   equ     13      ;disk reset
000E             100 selectf  equ     14      ;select disk
000F             101 openf    equ     15      ;open file
0010             102 closef   equ     16      ;close file
0013             103 delf     equ     19      ;delete file
0014             104 dreadf   equ     20      ;disk read
0016             105 makef    equ     22      ;make file
001A             106 dmaf     equ     26      ;set dma function
0019             107 curdrv   equ     25      ;get current drive
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ        SEQ         SOURCE STATEMENT

001F             108 dpbf    equ     31      ;get dpb address
0020             109 userf   equ     32      ;set/get user number
0025             110 resdvf  equ     37      ;reset drive
0031             111 scbf    equ     49      ;set/get system control block word
003C             112 rsxf    equ     60      ;RSX function call
0063             113 resalvf equ     99      ;reset allocation vector
006F             114 pblkf   equ     111     ;print block to console
0070             115 lblkf   equ     112     ;print block to list device
0080             116 ginitf  equ     128     ;GET initialization sub-function no.
0081             117 gkillf  equ     129     ;GET delete sub-function no.
0082             118 gfcbf   equ     130     ;GET file display sub-function no.
0084             119 pinitf  equ     132     ;PUT initialization sub-funct no.
0085             120 pckillf equ     133     ;PUT CON: delete sub-function no.
0086             121 pcfcbf  equ     134     ;return PUT CON: fcb address
0089             122 plkillf equ     137     ;PUT LST: delete sub-function no.
008A             123 plfcbf  equ     138     ;return PUT LST:fcb address
008C             124 jinitf  equ     140     ;JOURNAL initialization sub-funct no.
008D             125 jkillf  equ     141     ;JOURNAL delete sub-function no.
008E             126 jfcbf   equ     142     ;return JOURNAL fcb address
0090             127 skillf  equ     144     ;SUBMIT delete sub-function no.
0091             128 sfcbf   equ     145     ;SUBMIT fcb address function
00A0             129 svkillf equ     160     ;SAVE delete sub-function no.
                 130 ;
                 131 ;       System Control Block definitions
                 132 ;
003A             133 scba    equ     03ah    ;offset of scbadr from SCB base
00B3             134 ccpflg1 equ     0b3h    ;offset of ccpflags word from page boundary
0040             135 submit  equ     040h    ;mask for active submit or get test
00AA             136 errflg  equ     0aah    ;offset of error flag from page boundary
00CF             137 conmode equ     0cfh    ;offset of console mode from page boundary
00D4             138 listcp  equ     0d4h    ;offset of ^P flag from page boundary
00F9             139 common  equ     0f9h    ;offset of common memory base from pg. bound
0068             140 wbootfx equ     068h    ;offset of warm boot jmp from page. bound
006E             141 constfx equ     06eh    ;offset of constat jmp from page. bound
0074             142 coninfx equ     074h    ;offset of conin jmp from page. bound
007A             143 conoufx equ     07ah    ;offset of conout jmp from page. bound
0080             144 listfx  equ     080h    ;offset of list jmp from page. bound
0003             145 cstjmp  equ     003h    ;offset of console status jmp from warm boot
0006             146 cinjmp  equ     006h    ;offset of console input jmp from warm boot
0009             147 coujmp  equ     009h    ;offset of console output jmp from warm boot
000C             148 lstjmp  equ     00ch    ;offset of list output jmp from warm boot
                 149
                 150 ;
                 151 ;       Restore mode equates    (used with inr a, rz, rm, ret)
                 152 ;
00FF             153 norestore       equ     0ffh    ;no BIOS interception
007F             154 biosonly        equ     07fh    ;restore BIOS jump table only
0000             155 everything      equ     0       ;restore BIOS jump table and jmps in
                 156                                 ;RESBDOS (default mode)
                 157 ;
                 158 ;       Instructions
                 159 ;
0021             160 lxih            equ     21h     ;LXI H, instruction
00C3             161 jmpi            equ     0c3h    ;jump instruction
                 162 ;
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                  163 ;*****************************************************************
                  164 ;               START OF INITIALIZATION CODE
                  165 ;*****************************************************************
                  166                 cseg
                  167
                  168 putf:
                  169         ;get parameters
0000 60           170         mov     h,b
0001 69           171         mov     l,c             ;HL = .(parameter block)
0002 7E           172         mov     a,m             ;output type 0=con:,1=aux:,2=lst:,3=conin:
0003 FE01         173         cpi     1               ;is it aux?
0005 CA6701    C  174         jz      notimp          ;error if so
0008 FE03         175         cpi     3               ;is it console input only
000A C21100    C  176         jnz     setlst
000D 32C202    C  177         sta     input           ;non-zero => console input
0010 AF           178         xra     a
0011 32C302    C  179 setlst: sta     list            ;non-zero => list device
0014 23           180         inx     h
0015 7E           181         mov     a,m             ;echo/noecho mode
0016 32C402    C  182         sta     echo
0019 23           183         inx     h
001A 7E           184         mov     a,m             ;cooked/raw mode
001B 32C502    C  185         sta     cooked
001E 23           186         inx     h
001F 7E           187         mov     a,m             ;system/program mode
0020 32CA02    C  188         sta     program
                  189         ;
                  190         ;check if enough memory
                  191         ;
0023 2A0000    E  192         lhld    memsiz
0026 7C           193         mov     a,h
0027 FE20         194         cpi     20h
0029 113302    C  195         lxi     d,memerr
002C DA6A01    C  196         jc      error
                  197         ;
                  198         ;check if drive specified
002F 210000    E  199         lxi     h,fcb
0032 7E           200         mov     a,m             ;drive code
0033 3D           201         dcr     a               ;drive specified?
0034 F24000    C  202         jp      movfcb          ;jump if so
                  203         ;
                  204         ;set to current drive, if  not
                  205         ;
0037 0E19         206         mvi     c,curdrv
0039 E5           207         push    h               ;save .fcb
003A CD0000    E  208         call    bdos
003D E1           209         pop     h               ;a=current drive, hl=.fcb
003E 77           210         mov     m,a             ;set fcb to force drive select
003F 34           211         inr     m               ;must be relative to 1
                  212         ;
                  213 movfcb: ;copy default fcb up into data area for move to RSX
                  214         ;
0040 5F           215         mov     e,a
0041 0E0E         216         mvi     c,selectf       ;make sure drive is selected
0043 E5           217         push    h               ;save .fcb
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0044 CD0000   E   218         call    mon1            ;so we get the right DPB
0047 E1           219         pop     h
0048 11CC02   C   220         lxi     d,putfcb
004B 012000       221         lxi     b,32            ;length of fcb
004E CDAD01   C   222         call    ldir            ;move it to putfcb
                  223         ;
                  224         ;initialize other variables to be moved to RSX
                  225         ;
0051 CD9B01   C   226         call    getusr          ;get current user number
0054 32CB02   C   227         sta     putusr          ;save for redirection file I/O
0057 CDA301   C   228         call    getscbadr
005A 22A202   C   229         shld    scbadr          ;System Control Block address
                  230         ;
                  231         ;initialize records per block (BLM)
                  232         ;
005D 0E1F         233         mvi     c,dpbf
005F CD0000   E   234         call    mon1            ;HL = .disk parameter block
0062 23           235         inx     h
0063 23           236         inx     h
0064 23           237         inx     h               ;HL = .blm
0065 7E           238         mov     a,m
0066 32C902   C   239         sta     blm
.                 240         ;
                  241         ;initialize function table (functions to be intercepted)
                  242         ;
0069 3AC302   C   243         lda     list
006C B7           244         ora     a
006D 010D00       245         lxi     b,funcend-functbl       ;count
0070 11B502   C   246         lxi     d,functbl               ;destination
0073 218586       247         lxi     h,pcfcbf*256+pckillf    ;rsx function codes
0076 CA9400   C   248         jz      ckinput
0079 21B801   C   249         lxi     h,listfunc              ;list function table
007C CDAD01   C   250         call    ldir
007F 3E05         251         mvi     a,lchrf
0081 32B302   C   252         sta     bdosfunc                ;use list output for bios trap
0084 3E80         253         mvi     a,listfx
0086 329E02   C   254         sta     resoff                  ;offset of fixup for bios list
0089 3E0C         255         mvi     a,lst.jmp
008B 329F02   C   256         sta     biosoff                 ;offset of bios lst jmp
008E 21898A       257         lxi     h,plfcbf*256+plkillf
0091 C3C400   C   258         jmp     getrsxadr
                  259 ckinput:
0094 3AC202   C   260         lda     input
0097 B7           261         ora     a
0098 CAC400   C   262         jz      getrsxadr
009B 21C501   C   263         lxi     h,inputfunc
009E CDAD01   C   264         call    ldir
00A1 3E01         265         mvi     a,cinf
00A3 32B302   C   266         sta     bdosfunc                ;use console input
00A6 3E74         267         mvi     a,coninfx
00A8 329E02   C   268         sta     resoff                  ;offset of fixup for bios conin
00AB 3E06         269         mvi     a,cin.jmp
00AD 329F02   C   270         sta     biosoff
00B0 32C402   C   271         sta     echo                    ;must be non-zero for input
00B3 2AA202   C   272         lhld    scbadr
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER #

```
LOC  OBJ        SEQ        SOURCE STATEMENT

00B6 2EB4        273        mvi     l,ccpflg+1
00B8 7E          274        mov     a,m
00B9 E640        275        ani     submit
00BB 115202   C  276        lxi     d,noget
00BE C26A01   C  277        jnz     error                ;error if so
00C1 218D8E      278        lxi     h,jfcbf$256+jkillf
                 279        ;
                 280        ;get address of initialization table in RSX
                 281        ;
                 282 getrsxadr:
00C4 22C602   C  283        shld    rsxfun
00C7 0E3C        284        mvi     c,rsxf          ;PUT is not compatible with SAVE.RSX
00C9 11D201   C  285        lxi     d,savkill       ;as both SAVE & PUT trap warm starts
00CC CD0000   E  286        call    bdos            ;eliminate SAVE.RSX if active
00CF 0E3C        287        mvi     c,rsxf
00D1 11D301   C  288        lxi     d,rsxinit
00D4 CD0000   E  289        call    bdos            ;call PUT.RSX initialization routine
00D7 E5          290        push    h               ;save address of destination for move
00D8 5E          291        mov     e,m
00D9 23          292        inx     h
00DA 56          293        mov     d,m             ;DE = .kill flag
00DB D5          294        push    d               ;save for later set
                 295        ;
                 296 if biosfunctions
                 297        ;
00DC 23          298        inx     h
00DD 23          299        inx     h
00DE 23          300        inx     h               ;HL = .(.(bios entry in RSX))
00DF E5          301        push    h               ;save for getting RSX entry point
                 302                                ;later (in trap!)
                 303        ;check if BIOS jump table looks valid (jmp in right places)
00E0 2A9F02   C  304 check:  lhld    biosoff
00E3 EB          305        xchg
00E4 2A0100      306        lhld    wboota
00E7 7E          307        mov     a,m
00E8 FEC3        308        cpi     jmpi            ;should be a jump
00EA 19          309        dad     d               ;HL = .(jmp address)
00EB 7E          310        mov     a,m
00EC FEC3        311        cpi     jmpi            ;should be a jump
00EE C28A01   C  312        jnz     bioserr         ;skip bios redirection if not
                 313        ;
                 314        ;fix up RESBDOS to do BIOS calls to intercepted functions
                 315        ;
00F1 2AA202   C  316        lhld    scbadr
00F4 2EFA        317        mvi     l,common+1
00F6 7E          318        mov     a,m             ;get high byte of common base
00F7 B7          319        ora     a
00F8 C20301   C  320        jnz     fix0            ;high byte = zero if non-banked
00FB 3E7F        321        mvi     a,biosonly
00FD 32B402   C  322        sta     biosmode
0100 C32901   C  323        jmp     trap            ;skip code that fixes resbdos
                 324        ;fix warmboot BIOS jmp in resbdos
0103 2E68        325 fix0:   mvi     l,wbootfx       ;HL = .warm boot fix in SCB
0105 22B102   C  326        shld    wmfix           ;save for RSX restore at end
0108 7E          327        mov     a,m
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

0109 FEC3          328        cpi     jmpi            ;is it a jump instruction?
010B CA1301    C   329        jz      fix1            ;jump if so
010E 3E7F          330        mvi     a,biosonly      ;whoops already traped
0110 32B402    C   331        sta     biosmode
0113 36[illegible]     332 fix1:  mvi     m,lxih          ;change jump to an lxi h,
                   333        ;fix list bios jmp in resbdos
0115 3A9E02    C   334        lda     resoff
0118 6F            335        mov     l,a
0119 22AA02    C   336        shld    biosfix
011C 7E            337        mov     a,m
011D FEC3          338        cpi     jmpi            ;is it a jump instruction?
011F CA[illegible]01    C   339        jz      bioscK          ;jump if so
0122 3E7F          340        mvi     a,biosonly      ;whoops already changed
0124 32B402    C   341        sta     biosmode        ;restore jump table only
0127 3621          342 fix3:  mvi     m,lxih
                   343        ;
                   344        ;get address of list entry point
                   345        ;
0129 E1            346 trap:  pop     h               ;.(.(bios entry point in RSX))
012A 4E            347        mov     c,m
012B 23            348        inx     h
012C 46            349        mov     b,m
012D E5            350        push    h
012E 2A9F02    C   351        lhld    biosoff
0131 EB            352        xchg
0132 2A0100        353        lhld    wboota
0135 19            354        dad     d               ;HL = .(jmp address)
0136 23            355        inx     h               ;move past jmp instruction
0137 22A802    C   356        shld    biosjmp         ;save for RSX restore at end
013A 5E            357        mov     e,m
013B 71            358        mov     m,c
013C 23            359        inx     h
013D 56            360        mov     d,m             ;DE = bios routine address
013E 70            361        mov     m,b             ;BIOS jmp jumps to RSX
013F EB            362        xchg
0140 22A602    C   363        shld    biosout         ;save bios routine address
                   364        ;get addresses of RSX bios trap
0143 E1            365        pop     h
0144 23            366        inx     h
0145 4E            367        mov     c,m             ;HL = .(.(bios warm start in RSX))
0146 23            368        inx     h
0147 46            369        mov     b,m             ;BC = .bios warmstart entry in RSX
                   370        ;
                   371        ;patch RSX wmboot entry into BIOS jump table
                   372        ;save real wmboot address in RSX exit table
                   373        ;
0148 2A0100        374        lhld    wboota
014B 23            375        inx     h
014C 22AF02    C   376        shld    wmjmp           ;save for RSX restore at end
014F 5E            377        mov     e,m
0150 71            378        mov     m,c
0151 23            379        inx     h
0152 56            380        mov     d,m
0153 70            381        mov     m,b
0154 EB            382        xchg
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0155 22AD02   C   383         shld    wmsta           ;save real bios warm start routine
                  384 endif
                  385         ;
                  386         ;move data area to RSX
                  387         ;
                  388 rsxmov:
0158 E1           389         pop     h               ;HL = .(Kill flag = 0FFh)
0159 34           390         inr     m               ;set to zero for redirection active
015A 21A202   C   391         lxi     h,movstart
015D D1           392         pop     d               ;RSX data area address
015E 014C00       393         lxi     b,moveend-movstart
0161 CDAD01   C   394         call    ldir
0164 C30000       395         jmp     wboot
                  396 ;
                  397 ;       auxiliary redirection
                  398 ;
                  399 notimp:
0167 11FB01   C   400         lxi     d,notdone
                  401 error:
016A 0E09         402         mvi     c,pbuff
016C CD0000   E   403         call    mon1
016F 0E10         404         mvi     c,closef
0171 110000   E   405         lxi     d,fcb
0174 CD0000   E   406         call    mon1
0177 0E13         407         mvi     c,delf
0179 110000   E   408         lxi     d,fcb
017C CD0000   E   409         call    mon1
017F C30000       410         jmp     wboot
                  411
                  412
                  413 if biosfunctions
                  414 ;
                  415 ;       check if warm boot was fixed up by someone
                  416 ;       and list or console output was not
                  417 ;
0182 3AB402   C   418 biosck: lda     biosmode
0185 FE7F         419         cpi     biosonly
0187 C22701   C   420         jnz     fix3            ;warm boot not fixed up
                  421 ;
                  422 ;       can't do BIOS redirection
                  423 ;
                  424 bioserr:
018A 11D401   C   425         lxi     d,nobios
018D 0E09         426         mvi     c,pbuff
018F CD0000   E   427         call    mon1
0192 21B402   C   428         lxi     h,biosmode
0195 36FF         429         mvi     m,norestore
0197 E1           430         pop     h               ;throw away stacked bios entry
0198 C35801   C   431         jmp     rsxmov
                  432 endif
                  433 ;
                  434 ;       get/set user number
                  435 ;
019B 3EFF         436 getusr: mvi     a,0ffh          ;get current user number
019D 5F           437 setusr: mov     e,a             ;set current user number (in A)
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
LOC  OBJ          SEQ         SOURCE STATEMENT

019E 0E20          438         mvi     c,userf
01A0 C30000    E   439         jmp     mon1
                   440 ;
                   441 ;       get system control block address
                   442 ;       (BDOS function #49)
                   443 ;
                   444 ;       exit:   hl = system control block address
                   445 ;
                   446 getscbadr:
01A3 0E31          447         mvi     c,scbf
01A5 11AB01    C   448         lxi     d,data49
01A8 C30000    E   449         jmp     mon1
                   450 ;
01AB 3A            451 data49: db      scba,0          ;data structure for getscbadd
01AC 00
                   452 ;
                   453 ;
                   454 ;       copy memory bytes (emulates z80 ldir instruction)
                   455 ;
01AD 7E            456 ldir:   mov     a,m             ;get byte
01AE 12            457         stax    d               ;store it at destination
01AF 23            458         inx     h               ;advance pointers
01B0 13            459         inx     d
01B1 0B            460         dcx     b               ;decrement byte count
01B2 79            461         mov     a,c             ;loop if non-zero
01B3 B0            462         ora     b
01B4 C2AD01    C   463         jnz     ldir
01B7 C9            464         ret
                   465 ;
                   466 ;*****************************************************************
                   467 ;               DATA AREA
                   468 ;*****************************************************************
                   469
                   470 ;
                   471 ;       equates function table
                   472 ;
00FF               473 eot     equ     0ffh    ; end of function table
00FE               474 skipf   equ     0feh    ; skip this function
                   475 ;
                   476 listfunc:
01B8 05            477                 db      lchrf, lblkf, coutf, cstatf, crawf
01B9 70
01BA 02
01BB 0B
01BC 06
01BD 09            478                 db      pbuff, cinf, creadf, resetf, resdvf
01BE 01
01BF 0A
01C0 0D
01C1 25
01C2 63            479                 db      resalvf, pblkf, eot
01C3 6F
01C4 FF
                   480
                   481 ;               Note that the list routines precede the console
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

                  482 ;             routines so that the CKLIST: routine in PUTRSX
                  483 ;             can distinquish list functions from console
                  484 ;             functions.
                  485
                  486 inputfunc:                  ;preset for console input
 01C5 FE          487               db      skipf, skipf, skipf, skipf, crawf
 01C6 FE
 01C7 FE
 01C8 FE
 01C9 06
 01CA FE          488               db      skipf, cinf, creadf, resetf, resdvf
 01CB 01
 01CC 0A
 01CD 0D
 01CE 25
 01CF 63          489               db      resalvf, eot, skipf
 01D0 FF
 01D1 FE
                  490
                  491
                  492 ;
 01D2 A0          493 savkill:      db      svkillf
 01D3 84          494 rsxinit:      db      Pinitf
 01D4 0D          495 nobios:       db      cr,lf,'WARNING: Cannot redirect from BIOS',cr,lf,'$'
 01D5 0A
 01D6 5741524E
 01DA 494E473A
 01DE 2043616E
 01E2 6E6F7420
 01E6 72656469
 01EA 72656374
 01EE 2066726F
 01F2 6D204249
 01F6 4F53
 01F8 0D
 01F9 0A
 01FA 24
                  496 notdone:
 01FB 0D          497       db cr,lf
 01FC 0A
 01FD 4552524F    498       db 'ERROR: Auxiliary device redirection not implemented',cr,lf,'$'
 0201 523A2041
 0205 7578696C
 0209 69617279
 020D 20646576
 0211 69636520
 0215 72656469
 0219 72656374
 021D 696F6E20
 0221 6E6F7420
 0225 696D706C
 0229 656D656E
 022D 746564
 0230 0D
 0231 0A
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0232 24
                499 memerr:
0233 0D         500        db cr,lf
0234 0A
0235 4552524F   501        db 'ERROR: Insufficient Memory',cr,lf,'$'
0239 523A2049
023D 6E737566
0241 66696369
0245 656E7420
0249 4D656D6F
024D 7279
024F 0D
0250 0A
0251 24
                502 noget:
0252 0D         503        db cr,lf
0253 0A
0254 4552524F   504        db 'ERROR: You cannot PUT INPUT to a file',cr,lf
0258 523A2059
025C 6F752063
0260 616E6E6F
0264 74205055
0268 5420494E
026C 50555420
0270 746F2061
0274 2066696C
0278 65
0279 0D
027A 0A
027B 20202020   505        db '       when using GET or SUBMIT.',cr,lf,'$'
027F 20202077
0283 68656E20
0287 7573696E
028B 67204745
028F 54206F72
0293 20535542
0297 4D49542E
029B 0D
029C 0A
029D 24
029E 7A         506 resoff:        db      conoufx
029F 0900       507 biosoff:       dw      coujmp
02A1 00         508 aux:           db      0
                509        ;
                510 ;***************************************************************
                511 ;      Following variables are initialized by PUT.COM
                512 ;      and moved to the PUT RSX - Their order must not be changed
                513 ;***************************************************************
                514        ;
                515        ;
                516 movstart:
                517 inittable:                      ;addresses used by PUT.COM for
02A2 0000       518 scbadr: dw      0              ;address of System Control Block
                519        ;
                520        if biosfunctions        ;PUT.RSX initialization
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                521        ;
02A4 4B         522 gobios: mov     c,e
02A5 C3         523        db      jmpi
                524 biosout:
02A6 0000       525        dw      0               ;set to real BIOS routine
                526        ;
                527                                ;restore only if changed when removed.
                528 biosjmp:
02A8 0000       529        dw      0               ;address of bios jmp initialized by COM
                530 biosfix:
02AA 0000       531        dw      0               ;address of jmp in resbdos to restore
02AC C3         532        db      jmpi
02AD 0000       533 wmsta:  dw      0               ;address of real warm start routine
02AF 0000       534 wmjmp:  dw      0               ;address of jmp in bios to restore
02B1 0000       535 wmfix:  dw      0               ;address of jmp in resbdos to restore
                536 bdosfunc:
02B3 02         537        db      coutf
                538 biosmode:
02B4 00         539        db      0               ;0FFh = no bios restore, 07fh = restore
                540                                ;only bios jmp, 0 = restore bios jump and
                541                                ;resbdos jmp when removed.
                542        endif
                543
                544 functbl:                       ;preset for console output
02B5 FE         545        db      skipf, skipf, coutf, cstatf, crawf, pbuff
02B6 FE
02B7 02
02B8 0B
02B9 06
02BA 09
02BB 01         546        db      cinf, creadf, resetf, resdvf, resalvf, pblkf, eof
02BC 0A
02BD 0D
02BE 25
02BF 63
02C0 6F
02C1 FF
                547
                548 funcend:
                549        ;
02C2 00         550 input:  db      0               ;non-zero if putting input to a file
02C3 00         551 list:   db      0               ;TRUE if list output redirection
02C4 01         552 echo:   db      1               ;echo output to device
                553 cooked:                        ;must be next after echo
02C5 00         554        db      0               ;TRUE if ctrl chars displayed with ^
                555 rsxfun:
02C6 FF         556 pkillf: db      255             ;put abort routine code
02C7 FF         557 pfcbf:  db      255             ;put FCB display function no.
                558        ;       ********** remaining variables must be in this order
02C8 00         559 record: db      0               ;counts down records to block boundary
02C9 00         560 blm:    db      0               ;block mask = records per block (rel 0)
                561 program:                       ;This must be @ .putfcb-2
02CA 00         562        db      0
02CB 00         563 putusr: db      0               ;user number for redirection file
02CC 01         564 putfcb: db      1               ;a
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
  LOC  OBJ        SEQ         SOURCE STATEMENT

  02CD 5359534F    565         db      'SYSOUT  '
  02D1 55542020
  02D5 242424      566         db      '$$$'
  02D8 00          567         db      0,0
  02D9 00
  02DA 00          568 putmod: db      0
  02DB 00          569 putrc:  db      0
  0010             570         ds      16              ;map
  02EC 00          571 putcr:  db      0
                   572         ;
  02ED 00          573 cbufp:  db      0
                   574 movend:
                   575 ;*****************************************************************
                   576         end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

PUBLIC SYMBOLS
PUTF C 0000

EXTERNAL SYMBOLS
FCB E 0000 MEMSIZ E 0000 MON1 E 0000

USER SYMBOLS

AUX	C 02A1	BDOSFU	C 02B3	BIOSCK	C 0182	BIOSER	C 018A	BIOSFI	C 02AA	BIOSFU	A FFFF	BIOSJM	C 02A8
BIOSMO	C 02B4	BIOSOF	C 029F	BIOSON	A 007F	BIOSOU	C 02A6	BLM	C 02C9	CBUFP	C 02ED	CCPFLG	A 00B3
CHECK	C 00E0	CINF	A 0001	CINJMP	A 0006	CKINPU	C 0094	CLOSEF	A 0010	COMMON	A 00F9	CONINF	A 0074
CONMOD	A 00CF	CONOUF	A 007A	CONSTF	A 006E	COOKED	C 02C5	COUJMP	A 0009	COUTF	A 0002	CR	A 000D
CRAWF	A 0006	CREADF	A 000A	CSTATF	A 000B	CSTJMP	A 0003	CURDRV	A 0019	DATA49	C 01AB	DELF	A 0013
DMAF	A 001A	DPBF	A 001F	DREADF	A 0014	ECHO	C 02C4	EOT	A 00FF	ERRFLG	A 00AA	ERROR	C 016A
EVERYT	A 0000	FALSE	A 0000	FCB	E 0000	FIX0	C 0103	FIX1	C 0113	FIX3	C 0127	FUNCEN	C 02C2
FUNCTB	C 02B5	GETRSX	C 00C4	GETSCB	C 01A3	GETUSR	C 019B	GFCBF	A 0082	GINITF	A 0080	GKILLF	A 0081
GOBIOS	C 02A4	INITTA	C 02A2	INPUT	C 02C2	INPUTF	C 01C5	JFCBF	A 008E	JINITF	A 008C	JKILLF	A 008D
JMPI	A 00C3	LBLKF	A 0070	LCHRF	A 0005	LDIR	C 01AD	LF	A 000A	LIST	C 02C3	LISTCP	A 00D4
LISTFU	C 01B8	LISTFX	A 0080	LSTJMP	A 000C	LXIH	A 0021	MAKEF	A 0016	MEMERR	C 0233	MEMSIZ	E 0000
MON1	E 0000	MOVEND	C 02EE	MOVFCB	C 0040	MOVSTA	C 02A2	NOBIOS	C 01D4	NOGET	C 0252	NOREST	A 00FF
NOTDON	C 01FB	NOTIMP	C 0167	OPENF	A 000F	PBLKF	A 006F	PBUFF	A 0009	PCFCBF	A 0086	PCKILL	A 0085
PFCBF	C 02C7	PINITF	A 0084	PKILLF	C 02C6	PLFCBF	A 008A	PLKILL	A 0089	PROGRA	C 02CA	PUTCR	C 02EC
PUTF	C 0000	PUTFCB	C 02CC	PUTMOD	C 02DA	PUTRC	C 02DB	PUTUSR	C 02CB	RECORD	C 02C8	RESALV	A 0063
RESDVF	A 0025	RESETF	A 000D	RESOFF	C 029E	RSXF	A 003C	RSXFUN	C 02C6	RSXINI	C 01D3	RSXMOV	C 0158
SAVKIL	C 01D2	SCBA	A 003A	SCBADR	C 02A2	SCBF	A 0031	SELECT	A 000E	SETLST	C 0011	SETUSR	C 019D
SFCBF	A 0091	SKILLF	A 0090	SKIPF	A 00FE	SUBMIT	A 0040	SVKILL	A 00A0	TRAP	C 0129	TRUE	A FFFF
USERF	A 0020	WBOOT	A 0000	WBOOTA	A 0001	WBOOTF	A 0068	WMFIX	C 02B1	WMJMP	C 02AF	WMSTA	C 02AD

ASSEMBLY COMPLETE, NO ERRORS

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE PUT
OBJECT MODULE PLACED IN PUT.OBJ
COMPILER INVOKED BY:  PLM80 PUT.PLM XREF PAGEWIDTH(100) DEBUG OPTIMIZE


           $ TITLE('CP/M 3.0 --- PUT user interface')
  1        put:
           do;

           /*
             Copyright (C) 1982
             Digital Research
             P.O. Box 579
             Pacific Grove, CA 93950
           */

           /*
           Written:  02 Aug 82  by John Knight
           9/6/82  - changed RSX deletion & sub-function codes
                   - modified syntax & messages
                   - fixed password handling
           9/11/82 - sign-on message
           11/30/82 - interaction with SAVE
                    - PUT CONSOLE INPUT TO FILE
           */

           /*****************************************
           *                                        *
           *      LITERALS AND GLOBAL VARIABLES     *
           *                                        *
           *****************************************/

  2  1     declare
              true           literally '1',
              false          literally '0',
              forever        literally 'while true',
              lit            literally 'literally',
              proc           literally 'procedure',
              dcl            literally 'declare',
              addr           literally 'address',
              cr             literally '13',
              lf             literally '10',
              ctrlc          literally '3',
              ctrlx          literally '18h',
              bksp           literally '8',
              con$type       literally '0',
              aux$type       literally '1',
              list$type      literally '2',
              input$type         literally '3',
              con$width$offset   literally '1ah',
              ccp$flag$offset literally '18h',
              init$rsx       literally '132',
              Kill$con$rsx   literally '133',
              Kill$lst$rsx   literally '137',
              Kill$journal$rsx   literally '141',
              get$con$fcb    literally '134',
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                get$lst$fcb     literally '138',
                get$journal$fcb     literally '142',
                cpmversion      literally '30h';

  3   1         declare ccp$flag byte;
  4   1         declare con$width byte;
  5   1         declare i byte;
  6   1         declare begin$buffer address;
  7   1         declare buf$length byte;
  8   1         declare no$chars byte;
  9   1         declare rsx$kill$pb byte initial(kill$con$rsx);
 10   1         declare rsx$fcb$pb  byte initial(get$con$fcb);
 11   1         declare
                  warning (*)    byte data ('WARNING:',cr,lf,'$');

                /* scanner variables and data */
 12   1         declare
                  options(*) byte data
                      ('OUTPUT"TO"FILE"CONSOLE"CONOUT:"AUXILIARY"',
                       'AUXOUT:"END"CON:"AUX:"LIST"LST:"PRINTER"INPUT',0FFH),

                  options$offset(*) byte data
                      (0,7,10,15,23,31,41,49,53,58,63,68,73,81,86),

                  put$options(*) byte data
                      ('NOT"ECHO"RAW"FILTERED"SYSTEM"PROGRAM',0FFH),

                  put$options$offset(*) byte data
                      (0,4,9,13,22,29,36),

                  end$list    byte data (0ffh),

                  delimiters(*) byte data (0,'[]=, ./;',0,0ffh),

                  SPACE   byte data(5),

                  j       byte initial(0),
                  buf$ptr address,
                  index   byte,
                  endbuf  byte,
                  delimiter   byte;

 13   1           declare end$of$string  byte initial ('"');

 14   1         declare scbpd structure
                  (offset byte,
                   set    byte,
                   value  address);

 15   1         declare putpb structure
                  (output$type   byte,
                   echo$flag     byte,
                   filtered$flag byte,
                   program$flag   byte)
                   initial(con$type,true,true,true);

 16   1         declare parse$fn structure
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                (buff$adr address,
                 fcb$adr  address);

17   1      declare passwd (8) byte;

18   1      declare plm label public;

            /**************************************
             *                                    *
             *      B D O S   INTERFACE           *
             *                                    *
             **************************************/


19   1      mon1:
              procedure (func,info) external;
20   2          declare func byte;
21   2          declare info address;
22   2        end mon1;

23   1      mon2:
              procedure (func,info) byte external;
24   2          declare func byte;
25   2          declare info address;
26   2        end mon2;

27   1      mon3:
              procedure (func,info) address external;
28   2          declare func byte;
29   2          declare info address;
30   2        end mon3;

31   1      declare cmdrv     byte    external;  /* command drive      */
32   1      declare fcb (1)   byte    external;  /* 1st default fcb    */
33   1      declare fcb16 (1) byte    external;  /* 2nd default fcb    */
34   1      declare pass0     address external;  /* 1st password ptr   */
35   1      declare len0      byte    external;  /* 1st passwd length  */
36   1      declare pass1     address external;  /* 2nd password ptr   */
37   1      declare len1      byte    external;  /* 2nd passwd length  */
38   1      declare tbuff (1) byte    external;  /* default dma buffer */

            /**************************************
             *                                    *
             *      B D O S   Externals           *
             *                                    *
             **************************************/

39   1      read$console:
              procedure byte;
40   2          return mon2(1,0);
41   2        end read$console;

42   1      printchar:
              procedure(char);
43   2          declare char byte;
44   2          call mon1(2,char);
45   2        end printchar;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 46  1        conin:
                procedure byte;
 47  2          return mon2(6,0fdh);
 48  2          end conin;

 49  1        print$buf:
                procedure (buffer$address);
 50  2            declare buffer$address address;
 51  2            call mon1 (9,buffer$address);
 52  2          end print$buf;

 53  1        read$console$buf:
                procedure (buffer$address,max) byte;
 54  2          declare buffer$address address;
 55  2          declare new$max based buffer$address address;
 56  2          declare max byte;
 57  2          new$max = max;
 58  2          call mon1(10,buffer$address);
 59  2          buffer$address = buffer$address + 1;
 60  2          return new$max; /* actually number of characters input */
 61  2        end read$console$buf;

 62  1        version: procedure address;
                /* returns current cp/m version # */
 63  2          return mon3(12,0);
 64  2          end version;

 65  1        check$con$stat: procedure byte;
 66  2          return mon2(11,0);
 67  2        end check$con$stat;

 68  1        delete$file:
                procedure (fcb$address) address;
 69  2          declare fcb$address address;
 70  2          return mon3(19,fcb$address);
 71  2        end delete$file;

 72  1        make$file: procedure (fcb) address;
 73  2          declare fcb address;
 74  2          return mon3(22,fcb);
 75  2        end make$file;

 76  1        set$dma: procedure(dma);
 77  2          declare dma address;
 78  2          call mon1(26,dma);
 79  2        end set$dma;

                /* 0ffh ==> return BDOS errors */
 80  1        return$errors: procedure (mode);
 81  2          declare mode byte;
 82  2          call mon1(45,mode);
 83  2        end return$errors;

 84  1        getscbbyte: procedure (offset) byte;
 85  2          declare offset byte;
 86  2          scbpd.offset = offset;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 87  2          scbpd.set = 0;
 88  2          return mon2(49,.scbpd);
 89  2        end getscbbyte;

 90  1        setscbbyte:
              procedure (offset,value);
 91  2          declare offset byte;
 92  2          declare value byte;
 93  2          scbpd.offset = offset;
 94  2          scbpd.set = 0ffh;
 95  2          scbpd.value = double(value);
 96  2          call mon1(49,.scbpd);
 97  2        end setscbbyte;

 98  1      rsx$call: procedure (rsxpb) address;
         /* call Resident System Extension */
 99  2        declare rsxpb address;
100  2        return mon3(60,rsxpb);
101  2      end rsx$call;


102  1      get$console$mode: procedure address;
         /* returns console mode */
103  2        return mon3(6dh,0ffffh);
104  2      end get$console$mode;

105  1      set$console$mode: procedure (new$value);
106  2        declare new$value address;
107  2        call mon1(6dh,new$value);
108  2      end set$console$mode;

109  1      parse: procedure (pfcb) address external;
110  2        declare pfcb address;
111  2      end parse;

112  1      putf: procedure (param$block) external;
113  2        declare param$block address;
114  2      end putf;

          /****************************************
           *                                      *
           *      S U B R O U T I N E S           *
           *                                      *
           ****************************************/


         /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                        * * *  Option scanner  * * *

          * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


115  1      separator: procedure(character) byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        /* determines if character is a
                           delimiter and which one */
116  2      declare k   byte,
               character   byte;

117  2      k = 1;
118  2  loop:  if delimiters(k) = end$list then return(0);
120  2      if delimiters(k) = character then return(k);    /* null = 25 */
122  2          k = k + 1;
123  2          go to loop;

124  2  end separator;

125  1  opt$scanner:    procedure(list$ptr,off$ptr,idx$ptr);
                        /* scans the list pointed at by idxptr
                           for any strings that are in the
                           list pointed at by list$ptr.
                           Offptr points at an array that
                           contains the indices for the known
                           list. Idxptr points at the index
                           into the list. If the input string
                           is unrecognizable then the index is
                           0, otherwise > 0.

                           First, find the string in the known
                           list that starts with the same first
                           character.  Compare up until the next
                           delimiter on the input. if every input
                           character matches then check for
                           uniqueness.  Otherwise try to find
                           another known string that has its first
                           character match, and repeat.  If none
                           can be found then return invalid.

                           To test for uniqueness, start at the
                           next string in the known list and try
                           to get another match with the input.
                           If there is a match then return invalid.

                           else move pointer past delimiter and
                           return.

                       P.Balma     */

126  2      declare
               buff        based buf$ptr (1) byte,
               idx$ptr     address,
               off$ptr     address,
               list$ptr    address;

127  2      declare
               i       byte,
               j       byte,
               list        based list$ptr (1) byte,
               offsets     based off$ptr (1) byte,
               wrd$pos     byte,
               character   byte,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                letter$in$word  byte,
                found$first byte,
                start       byte,
                index       based idx$ptr byte,
                save$index  byte,
                (len$new,len$found) byte,
                valid       byte;

        /*****************************************************************************/
        /*          internal subroutines                                             */
        /*****************************************************************************/

128  2  check$in$list: procedure;
                    /* find known string that has a match with
                       input on the first character.  Set index
                       = invalid if none found.   */

129  3      declare i   byte;

130  3      i = start;
131  3      wrd$pos = offsets(i);
132  3      do while list(wrd$pos) <> end$list;
133  4          i = i + 1;
134  4          index = i;
135  4          if list(wrd$pos) = character then return;
137  4          wrd$pos = offsets(i);
138  4      end;
                    /* could not find character */
139  3      index = 0;
140  3      return;
141  3  end check$in$list;

142  2  setup:  procedure;
143  3      character = buff(0);
144  3      call check$in$list;
145  3      letter$in$word = wrd$pos;
                    /* even though no match may have occurred, position
                       to next input character.  */
146  3      i = 1;
147  3      character = buff(1);
148  3  end setup;

149  2  test$letter:    procedure;
                    /* test each letter in input and known string */

150  3      letter$in$word = letter$in$word + 1;

                        /* too many chars input? 0 means
                           past end of known string */
151  3      if list(letter$in$word) = end$of$string then valid = false;
           else
153  3      if list(letter$in$word) <> character then valid = false;

           i = i + 1;
156  3      character = buff(i);

157  3  end test$letter;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
158  2     skip:  procedure;
                          /* scan past the offending string;
                             position buf$ptr to next string...
                             skip entire offending string;
                             ie., falseopt=mod, [note: comma or
                             space is considered to be group
                             delimiter] */
159  3         character = buff(i);
160  3         delimiter = separator(character);
               /* No skip for PUT */
161  3             do while ((delimiter < 1) or (delimiter > 9));
162  4             i = i + 1;
163  4             character = buff(i);
164  4             delimiter = separator(character);
165  4         end;
166  3         endbuf = i;
167  3         buf$ptr = buf$ptr + endbuf + 1;
168  3         return;
169  3     end skip;

170  2     eat$blanks: procedure;

171  3         declare charac  based buf$ptr byte;

172  3         do while ((delimiter := separator(charac)) = SPACE);
173  4             buf$ptr = buf$ptr + 1;
174  4         end;

175  3     end eat$blanks;

          /*****************************************************************************/
          /*          end of internals                                                 */
          /*****************************************************************************/


                          /* start of procedure */
176  2         if delimiter = 9 then
177  2             return;         /* return if at end of buffer */
178  2         call eat$blanks;
179  2         start = 0;
180  2         call setup;

                          /* match each character with the option
                             for as many chars as input
                             Please note that due to the array
                             indices being relative to 0 and the
                             use of index both as a validity flag
                             and as a index into the option/mods
                             list, index is forced to be +1 as an
                             index into array and 0 as a flag*/

181  2         do while index <> 0;
182  3             start = index;
183  3             delimiter = separator(character);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        /* check up to input delimiter */

184  3          valid = true;       /* test$letter resets this */
185  3          do while delimiter = 0;
186  4             call test$letter;
187  4             if not valid then go to exit1;
189  4             delimiter = separator(character);
190  4          end;

191  3          go to good;

                        /* input ^= this known string;
                           get next known string that
                           matches */
192  3     exit1:    call setup;
193  3        end;
                        /* fell through from above, did
                           not find a good match*/
194  2        endbuf = i;        /* skip over string & return*/
195  2        call skip;
196  2        return;

                        /* is it a unique match in options
                           list? */
197  2     good:  endbuf = i;
198  2        len$found = endbuf;
199  2        save$index = index;
200  2        valid = false;
201  2     next$opt:
              start = index;
202  2          call setup;
203  2          if index = 0 then go to finished;

                        /* look at other options and check
                           uniqueness */

205  2          len$new = offsets(index + 1) - offsets(index) - 1;
206  2          if len$new = len$found then do;
208  3             valid = true;
209  3             do j = 1 to len$found;
210  4                call test$letter;
211  4                if not valid then go to next$opt;
213  4             end;
214  3          end;
215  2          else go to nextopt;
                        /* fell through...found another valid
                           match --> ambiguous reference */
216  2        index = 0;
217  2        call skip;      /* skip input field to next delimiter*/
218  2        return;

219  2     finished:          /* unambiguous reference */
             index = save$index;
220  2        buf$ptr = buf$ptr + endbuf;
221  2        call eat$blanks;
222  2        if delimiter <> 0 then
223  2             buf$ptr = buf$ptr + 1;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                    else
224   2                   delimiter = 5;
225   2             return;

226   2         end opt$scanner;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

227   1         crlf:  proc;
228   2             call printchar(cr);
229   2             call printchar(lf);
230   2             end crlf;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                /* fill string @ s for c bytes with f */
231   1         fill: procedure(s,f,c);
232   2           declare s address;
233   2           declare (f,c) byte;
234   2           declare a based s byte;
235   2           do while (c:=c-1) <> 255;
236   3             a=f;
237   3             s=s+1;
238   3           end;
239   2         end fill;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                /* The error processor.  This routine prints the command line
                   with a carot '^' under the offending delimiter, or sub-string.
                   The code passed to the routine determines the error message
                   to be printed beneath the command string.                  */

240   1         error: procedure (code);
241   2           declare (code,i,j,nlines,rem) byte;
242   2           declare (string$ptr,tstring$ptr) address;
243   2           declare chr1 based string$ptr byte;
244   2           declare chr2 based tstring$ptr byte;
245   2           declare carot$flag byte;

246   2         print$command: procedure (size);
247   3           declare size byte;
248   3           do j=1 to size;   /* print command string */
249   4             call printchar(chr1);
250   4             string$ptr = string$ptr + 1;
251   4           end;
252   3           call crlf;
253   3           do j=1 to size;   /* print carot if applicable */
254   4             if .chr2 = buf$ptr then do;
256   5               carot$flag = true;
257   5               call printchar('^');
258   5             end;
                    else
259   4               call printchar(' ');
260   4             tstring$ptr = tstring$ptr + 1;
261   4           end;
262   3           call crlf;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
263  3     end print$command;

264  2       carot$flag = false;
265  2       string$ptr,tstring$ptr = begin$buffer;
266  2       con$width = getscbbyte(con$width$offset);
267  2       if con$width < 40 then con$width = 40;
269  2       nlines = buf$length / con$width;  /* num lines to print */
270  2       rem = buf$length mod con$width;   /* num extra chars to print */
271  2       if code <> 2 then do;
273  3         if ((code = 1) or (code = 4)) then  /* adjust carot pointer */
274  3           buf$ptr = buf$ptr - 1;    /* for delimiter errors */
275  3         else if code <> 5 then
276  3           buf$ptr = buf$ptr - endbuf - 1;   /* all other errors */
                end;
278  2       call crlf;
279  2       do i=1 to nlines;
280  3         tstring$ptr = string$ptr;
281  3         call print$command(con$width);
282  3       end;
283  2       call print$command(rem);
284  2       if carot$flag then
285  2         call print$buf(.('Error at the ''^''; $'));
                else
286  2         call print$buf(.('Error at end of line; $'));
287  2       if con$width < 65 then
288  2         call crlf;
289  2       do case code;
290  3         call print$buf(.('Invalid option or modifier$'));
291  3         call print$buf(.('End of line expected$'));
292  3         call print$buf(.('Invalid file specification$'));
293  3         call print$buf(.('Invalid command$'));
294  3         call print$buf(.('Invalid delimiter$'));
295  3         call print$buf(.('File is Read Only$'));
296  3       end;
297  2       call mon1(0,0);
298  2     end error;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

299  1     user$abort: procedure (a);
300  2       declare a address;
301  2       declare response byte;

302  2       call print$buf(a);
303  2       call print$buf(.(' (Y/N)? $'));
304  2       response=read$console;
305  2       call crlf;
306  2       if not((response='y') or (response='Y')) then do;
308  3           call print$buf(.('PUT aborted$'));
309  3           call mon1(0,0);
310  3         end;
311  2       end user$abort;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

312  1     ucase: procedure (char) byte;
313  2       declare char byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
314  2      if char >= 'a' then
315  2        if char < '{' then
316  2          return (char-20h);
317  2      return char;
318  2    end ucase;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

319  1    getucase: procedure byte;
320  2      declare c byte;
321  2      c = ucase(conin);
322  2      return c;
323  2    end getucase;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

324  1    getpasswd: procedure;
325  2      declare (i,c) byte;
326  2      call crlf;
327  2      call crlf;
328  2      call print$buf(.('Enter Password: $'));
329  2    retry:
           call fill(.passwd,' ',8);
330  2      do i=0 to 7;
331  3    nxtchr:
           if (c:=getucase) >= ' ' then
332  3        passwd(i)=c;
333  3      if c = cr then
334  3        return;
335  3      if c = ctrlx then
336  3        go to retry;
337  3      if c = bksp then do;
339  4        if i < 1 then
340  4          goto retry;
341  4        else do;
342  5          passwd(i := i - 1) = ' ';
343  5          goto nxtchr;
344  5          end;
345  4        end;
346  3      if c = 3 then
347  3        call mon1(0,0);
348  3      end;
349  2    end getpasswd;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

350  1    put$msg: procedure;
351  2      call print$buf(.('Putting $'));
352  2      if putpb.output$type = list$type then
353  2        call print$buf(.('list$'));
           else
354  2        call print$buf(.('console$'));
355  2      if putpb.output$type = input$type then
356  2        call print$buf(.(' input to $'));
           else
357  2        call print$buf(.(' output to $'));
358  2    end put$msg;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

359   1         print$fn: procedure (fcb$ad);
360   2           declare K byte;
361   2           declare fcb$ad address;
362   2           declare driv based fcb$ad byte;
363   2           declare fn based fcb$ad (12) byte;

364   2           if getscbbyte(26) < 48 then
365   2             call crlf;              /* console width */
366   2           call print$buf(.('file: $'));
367   2           if driv <> 0 then do;
369   3             call printchar('@'+driv);
370   3             call printchar(':');
371   3           end;
372   2           do K=1 to 11;
373   3             if K=9 then
374   3               call printchar('.');
375   3             if fn(K) <> ' ' then
376   3               call printchar(fn(K));
377   3           end;
378   2         end print$fn;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

379   1         try$open: procedure;
380   2           declare (error$code,a) address;
381   2           declare prog$flag based a byte;
382   2           declare code byte;

383   2           error$code = rsx$call(.rsx$fcb$pb);
384   2           if error$code <> 0ffh then do;    /* ff means no active PUT file */
386   3             a = error$code - 2;         /* program output only? */
387   3             if prog$flag then
388   3               a = rsx$call(.rsx$kill$pb);       /* kill it if so */
389   3             else do;
390   4               call print$buf(.warning);
391   4               call put$msg;
392   4               call print$fn(error$code);    /* print the file name */
393   4               call user$abort(.(cr,lf,'Do you want another file$'));
394   4             end;
395   3           end;

396   2           call return$errors(0ffh);
397   2           call setdma(.passwd); /* set dma to password */
398   2           if passwd(0) <> ' ' then
399   2             fcb(6) = fcb(6) or 80h;
400   2           error$code=make$file(.fcb);
401   2           if low(error$code)=0ffh then do;  /* make failed? */
403   3             code = high(error$code);
404   3             if code = 8 then do;    /* file already exists */
406   4               call print$buf(.warning);
407   4               call user$abort(.('File already exists; Delete it$'));
408   4               error$code = delete$file(.fcb);
409   4               if low(error$code) = 0ffh then do;
411   5                 code = high(error$code);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
412  5            if code = 3 then    /* file is read only */
413  5              call error(5);
414  5            if code = 7 then do;    /* Password protected */
416  6              call getpasswd;
417  6              call crlf;
418  6            end;
419  5            call return$errors(0);
420  5            error$code=delete$file(.fcb);
421  5          end;
422  4        end;
423  3        call return$errors(0);
424  3        if passwd(0) <> ' ' then
425  3          fcb(6) = fcb(6) or 80h;
426  3        error$code = make$file(.fcb);
427  3      end;
428  2      call return$errors(0);
429  2      call put$msg;
430  2      call print$fn(.fcb);      /* print the file name */
431  2      call putf(.putpb);    /* do PUT processing */
       /*call mon1(0,0);      debug exit */
432  2    end try$open;

       /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

433  1    Kill$rsx: procedure;
434  2      declare (fcb$adr,a) address;

435  2      if (delimiter <> 9) and (delimiter <> 2) then /* check for eoln or ']' */
436  2        call error(1);
       /* remove PUT RSX */
437  2      do while (fcb$adr:=rsx$call(.rsx$fcb$pb)) <> 0ffh;
438  3        a = rsx$call(.rsx$kill$pb);
439  3        call print$buf(.('PUT completed for $'));
440  3        call print$fn(fcb$adr);
441  3        call crlf;
442  3      end;
443  2      call put$msg;
444  2      if putpb.output$type = list$type then
445  2        call print$buf(.('printer$'));
       else
446  2        call print$buf(.('console$'));
447  2      call mon1(0,0);
448  2    end kill$rsx;

       /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

449  1    output$options: procedure;
450  2      declare negate byte;
451  2      do while ((delimiter<>2) and (delimiter<>9));
452  3        negate = false;
453  3        call opt$scanner(.put$options(0),.put$options$offset(0),.index);
454  3        if index = 1 then do;   /* NOT */
456  4          negate = true;
457  4          call opt$scanner(.put$options(0),.put$options$offset(0),.index);
458  4        end;
459  3        if (index=0) or (index=1) then
460  3          call error(0);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
461  3          if index = 2 then do;   /* ECHO */
463  4            if negate then
464  4              putpb.echo$flag = false;
                  else
465  4              putpb.echo$flag = true;
466  4          end;
467  3          if index = 3 then do;   /* RAW output */
469  4            if negate then
470  4              putpb.filtered$flag = true;
                  else
471  4              putpb.filtered$flag = false;
472  4          end;
473  3          if index = 4 then do;   /* FILTERED output */
475  4            if negate then
476  4              putpb.filtered$flag = false;
                  else
477  4              putpb.filtered$flag = true;
478  4          end;
479  3          if index = 5 then do;   /* SYSTEM output */
481  4            if negate then
482  4              putpb.program$flag = true;
                  else
483  4              putpb.program$flag = false;
484  4          end;
485  3          if index = 6 then do;   /* PROGRAM output */
487  4            if negate then
488  4              putpb.program$flag = false;
                  else
489  4              putpb.program$flag = true;
490  4          end;
491  3        end;
492  2      end output$options;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

493  1      process$file: procedure(buf$adr);
494  2        declare status address;
495  2        declare buf$adr address;
496  2        declare char based status byte;
497  2        parse$fn.buff$adr = buf$adr;
498  2        parse$fn.fcb$adr = .fcb;
499  2        status = parse(.parse$fn);
500  2        if status = 0ffffh then do;
502  3          buf$ptr = parse$fn.buff$adr;
503  3          call error(2);  /* bad file */
504  3        end;
505  2        call move(8,.fcb16,.passwd);
506  2        if status = 0 then    /* eoln */
507  2          call try$open;
508  2        else do;
509  3          buf$ptr = status + 1;   /* position buf$ptr past '[' */
510  3          if char <> '[' then
511  3            call error(4);    /* Invalid delimiter */
512  3          else do;
513  4            call output$options;  /* process output options */
514  4            call try$open;
515  4          end;
```

CP/M - PLUS V 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
516  3       end;
517  2     end process$file;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

518  1     input$found: procedure (buffer$adr) byte;
519  2       declare buffer$adr address;
520  2       declare char based buffer$adr byte;
521  2       do while (char = ' ') or (char = 9); /* tabs & spaces */
522  3         buffer$adr = buffer$adr + 1;
523  3       end;
524  2       if char = 0 then  /* eoln */
525  2         return false;   /* input not found */
             else
526  2         return true;    /* input found */
527  2     end input$found;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

            /**********************************
           *                                 *
           *   M A I N   P R O G R A M       *
           *                                 *
           **********************************/

528  1     plm:
           do;
529  2       if (low(version) < cpmversion) or (high(version)=1) then do;
531  3         call print$buf(.('Requires CP/M 3.0$'));
532  3         call mon1(0,0);
533  3       end;
           /* default modes for putf call */
534  2       if not input$found(.tbuff(1)) then do;  /* just PUT, no command tail */
536  3           call print$buf(.('CP/M 3 PUT Version 3.0',cr,lf,'$'));
537  3           call print$buf(.('Put console output to a file$'));
538  3           call print$buf(.(cr,lf,'Enter file: $'));
539  3           no$chars = read$console$buf(.tbuff(0),128);
540  3           call crlf;
541  3           tbuff(1) = ' ';     /* blank out nc field */
542  3           tbuff(no$chars+2) = 0;  /* mark eoln */
543  3           if not input$found(.tbuff(1)) then  /* quit, no file name */
544  3             call mon1(0,0);
545  3           do i=1 to no$chars; /* make input capitals */
546  4             tbuff(i+1) = ucase(tbuff(i+1));
547  4           end;
548  3           begin$buffer = .tbuff(2);
549  3           buf$length = no$chars;
550  3           buf$ptr = .tbuff(2);
551  3           call process$file(.tbuff(2));
552  3       end;
553  2       else do;    /* Put with input */
554  3         i = 1;            /* skip over leading spaces */
555  3         do while (tbuff(i) = ' ');
556  4           i = i + 1;
557  4         end;
558  3         begin$buffer = .tbuff(1); /* note beginning of input */
559  3         buf$length = tbuff(0);    /* note length of input */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
560  3          buf$ptr = .tbuff(i);  /* set up for scanner */
561  3          index = 0;
562  3          delimiter = 1;
563  3          call opt$scanner(.options(0),.options$offset(0),.index);
564  3          if (index=6) or (index=7) or (index=10) then do;  /* AUX: */
566  4            putpb.output$type = aux$type;
567  4            call opt$scanner(.options(0),.options$offset(0),.index);
568  4            if index = 1 then   /* OUTPUT */
569  4              call opt$scanner(.options(0),.options$offset(0),.index);
570  4            if index = 2 then   /* TO */
571  4              call opt$scanner(.options(0),.options$offset(0),.index);
572  4            if index = 3 then   /* FILE */
573  4              call process$file(buf$ptr);
574  4            else do;
575  5              if (index=6) or (index=7) or (index=10) then  /* AUX: */
576  5                call kill$rsx;
                   else
577  5                call error(3);
578  5            end;
579  4          end;
580  3          else do;  /* not AUX, check LST */
581  4            if (index=11) or (index=12) or (index=13) then do;  /* LST: */
583  5              putpb.output$type = list$type;
584  5              putpb.echo$flag = false;  /* don't echo list output */
585  5              rsx$fcb$pb = get$lst$fcb;
586  5              rsx$kill$pb = kill$lst$rsx;
587  5              call opt$scanner(.options(0),.options$offset(0),.index);
588  5              if index = 1 then /* OUTPUT */
589  5                call opt$scanner(.options(0),.options$offset(0),.index);
590  5              if index = 2 then /* TO */
591  5                call opt$scanner(.options(0),.options$offset(0),.index);
592  5              if index = 3 then /* FILE */
593  5                call process$file(buf$ptr);
594  5              if (index=11) or (index=12) or (index=13) then    /* LIST */
595  5                call kill$rsx;
                     else
596  5                call error(3);
597  5            end;
598  4            else do;    /* normal CONSOLE output */
                     /* if CONSOLE or CONOUT or CON: */
599  5              if (index=4) or (index=5) or (index=9) then do;    /* CONSOLE */
601  6                if delimiter = 9 then
602  6                  call kill$rsx;
                       else
603  6                  call opt$scanner(.options(0),.options$offset(0),.index);
604  6                end;
605  5              if index = 1 then       /* OUTPUT */
606  5                call opt$scanner(.options(0),.options$offset(0),.index);
607  5              else if index = 14 then do; /* INPUT  */
609  6                putpb.output$type = input$type;
610  6                putpb.echo$flag = true;
611  6                putpb.filtered$flag = false;
612  6                rsx$fcb$pb = get$journal$fcb;
613  6                rsx$kill$pb = kill$journal$rsx;
614  6                call opt$scanner(.options(0),.options$offset(0),.index);
615  6              end;
                   if index = 2 then       /* TO */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
617  5              call opt$scanner(.options(0),.options$offset(0),.index);
618  5            if index = 3 then       /* FILE */
619  5              call process$file(buf$ptr);
620  5            if (index=4) or (index=5) or (index=9) then /* CONOUT: or CONSOLE */
621  5              call kill$rsx;
                    else
622  5              call error(3);
623  5          end;
624  4        end;
625  3      end;
626  2    end;
627  1  end put;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
434	006CH	2	A ADDRESS 438
299	005BH	2	A ADDRESS PARAMETER 300 302
234	0000H	1	A BYTE BASED(S) 236
380	0067H	2	A ADDRESS 381 386 388
2			ADDR LITERALLY
2			AUXTYPE LITERALLY 566
6	0003H	2	BEGINBUFFER ADDRESS 265 548 558
2			BKSP LITERALLY 337
493	006FH	2	BUFADR ADDRESS PARAMETER 495 497
126	0000H	1	BUFF BYTE BASED(BUFPTR) ARRAY(1) 143 147 156 159 163
16	0000H	2	BUFFADR ADDRESS MEMBER(PARSEFN) 497 502
49	0025H	2	BUFFERADDRESS ADDRESS PARAMETER 50 51
53	0027H	2	BUFFERADDRESS ADDRESS PARAMETER 54 55 58 59
518	0073H	2	BUFFERADR ADDRESS PARAMETER 519 520 522
7	0005H	1	BUFLENGTH BYTE 269 270 549 559
12	000AH	2	BUFPTR ADDRESS 126 167 171 173 220 223 254 274 276 502 509 550 560 573 593 619
231	004FH	1	C BYTE PARAMETER

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                    233  235

  325 0061H     1  C                BYTE
                                    331  332  333  335  337  346

  320 005FH     1  C                BYTE
                                    321  322

  245 0059H     1  CAROTFLAG        BYTE
                                    256  264  284

    3 0000H     1  CCPFLAG          BYTE

    2             CCPFLAGOFFSET    LITERALLY

   42 0024H     1  CHAR             BYTE PARAMETER
                                     43   44

  520 0000H     1  CHAR             BYTE BASED(BUFFERADR)
                                    521  524

  312 005EH     1  CHAR             BYTE PARAMETER
                                    313  314  315  316  317

  496 0000H     1  CHAR             BYTE BASED(STATUS)
                                    510

  171 0000H     1  CHARAC           BYTE BASED(BUFPTR)
                                    172

  115 0038H     1  CHARACTER        BYTE PARAMETER
                                    116  120

  127 0043H     1  CHARACTER        BYTE
                                    135  143  147  153  156  159  160  163  164  183  189

   65 05F7H     9  CHECKCONSTAT     PROCEDURE BYTE STACK=0002H

  128 0809H    86  CHECKINLIST      PROCEDURE STACK=0000H
                                    144

  243 0000H     1  CHR1             BYTE BASED(STRINGPTR)
                                    249

  244 0000H     1  CHR2             BYTE BASED(TSTRINGPTR)
                                    254

   31 0000H     1  CMDRV            BYTE EXTERNAL(3)

  240 0050H     1  CODE             BYTE PARAMETER
                                    241  271  273  275  289

  382 0069H     1  CODE             BYTE
                                    403  404  411  412  414

   46 05AEH     9  CONIN            PROCEDURE BYTE STACK=0002H
                                    321
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
     2             CONTYPE            LITERALLY
                                       15

     4 0001H   1   CONWIDTH           BYTE
                                       266  267  268  269  270  281  287

     2             CONWIDTHOFFSET     LITERALLY
                                       266

     2             CPMVERSION         LITERALLY
                                       529

     2             CR                 LITERALLY
                                       11  228  333  393  536  538

   227 0935H  11   CRLF               PROCEDURE STACK=0004H
                                       252  262  278  288  305  326  327  365  417  441  540

     2             CTRLC              LITERALLY

     2             CTRLX              LITERALLY
                                       335

     2             DCL                LITERALLY

    68 0600H  16   DELETEFILE         PROCEDURE ADDRESS STACK=0002H
                                       408  420

    12 000EH   1   DELIMITER          BYTE
                                       160  161  164  172  176  183  185  189  222  224  435
                                       451  562  601

    12 009EH  11   DELIMITERS         BYTE ARRAY(11) DATA
                                       118  120

    76 002EH   2   DMA                ADDRESS PARAMETER
                                       77  78

               DOUBLE                 BUILTIN
                                       95

   362 0000H   1   DRIV               BYTE BASED(FCBAD)
                                       367  369

   170 0919H  28   EATBLANKS          PROCEDURE STACK=0002H
                                       178  221

    15 0001H   1   ECHOFLAG           BYTE MEMBER(PUTPB)
                                       464  465  584  610

    12 000DH   1   ENDBUF             BYTE
                                       166  167  194  197  198  220  276

    12 009DH   1   ENDLIST            BYTE DATA
                                       118  132

    13 000FH   1   ENDOFSTRING        BYTE INITIAL
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                                151

240 096BH   305  ERROR           PROCEDURE STACK=0008H
                                 413  436  460  503  511  577  596  622

380 0065H     2  ERRORCODE       ADDRESS
                                 383  384  386  392  400  401  403  408  409  411  420
                                 426

192 0743H        EXIT1          LABEL
                                 188

231 004EH     1  F               BYTE PARAMETER
                                 233  236

  2              FALSE          LITERALLY
                                 152  154  200  264  452  464  471  476  483  488  525
                                 584  611

 32 0000H     1  FCB             BYTE ARRAY(1) EXTERNAL(4)
                                 399  400  408  420  425  426  430  498

 72 002CH     2  FCB             ADDRESS PARAMETER
                                  73   74

 33 0000H     1  FCB16           BYTE ARRAY(1) EXTERNAL(5)
                                 505

359 0062H     2  FCBAD           ADDRESS PARAMETER
                                 361  362  363

 68 002AH     2  FCBADDRESS      ADDRESS PARAMETER
                                  69   70

434 006AH     2  FCBADR          ADDRESS
                                 437  440

 16 0002H     2  FCBADR          ADDRESS MEMBER(PARSEFN)
                                 498

231 0940H    43  FILL            PROCEDURE STACK=0002H
                                 329

 15 0002H     1  FILTEREDFLAG    BYTE MEMBER(PUTPB)
                                 470  471  476  477  611

219 07D9H        FINISHED       LABEL
                                 204

363 0000H    12  FN              BYTE BASED(FCBAD) ARRAY(12)
                                 375  376

  2              FOREVER        LITERALLY

127 0045H     1  FOUNDFIRST      BYTE

 23 0000H     1  FUNC            BYTE PARAMETER
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                  24

 27 0000H     1  FUNC              BYTE PARAMETER
                                   28

 19 0000H     1  FUNC              BYTE PARAMETER
                                   20

  2               GETCONFCB        LITERALLY
                                   10

102 068AH     9  GETCONSOLEMODE    PROCEDURE ADDRESS STACK=0002H

  2               GETJOURNALFCB    LITERALLY
                                   612

  2               GETLSTFCB        LITERALLY
                                   585

324 0B77H   143  GETPASSWD         PROCEDURE STACK=0006H
                                   416

 84 0640H    24  GETSCBBYTE        PROCEDURE BYTE STACK=0002H
                                   266  364

319 0B69H    14  GETUCASE          PROCEDURE BYTE STACK=0004H
                                   331

197 0753H        GOOD              LABEL
                                   191

                 HIGH              BUILTIN
                                   403  411  529

  5 0002H     1  I                 BYTE
                                   545  546  554  555  556  560

325 0060H     1  I                 BYTE
                                   330  332  339  342

129 004BH     1  I                 BYTE
                                   130  131  133  134  137

127 0040H     1  I                 BYTE
                                   146  155  156  159  162  163  166  194  197

241 0051H     1  I                 BYTE
                                   279

125 003EH     2  IDXPTR            ADDRESS PARAMETER
                                   126  127

127 0000H     1  INDEX             BYTE BASED(IDXPTR)
                                   134  139  181  182  199  201  203  205  216  219

 12 000CH     1  INDEX             BYTE
                                   453  454  457  459  461  467  473  479  485  561  563
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                564  567  568  569  570  571  572  575  581  587  588
                                589  590  591  592  594  599  603  605  606  607  614
                                616  617  618  620

  23 0000H     2  INFO          ADDRESS PARAMETER
                                 25

  27 0000H     2  INFO          ADDRESS PARAMETER
                                 29

  19 0000H     2  INFO          ADDRESS PARAMETER
                                 21

   2              INITRSX       LITERALLY

 518 0F7DH    58  INPUTFOUND    PROCEDURE BYTE STACK=0002H
                                534  543

   2              INPUTTYPE     LITERALLY
                                355  609

 241 0052H     1  J             BYTE
                                248  253

 127 0041H     1  J             BYTE
                                209

  12 0009H     1  J             BYTE INITIAL

 360 0064H     1  K             BYTE
                                372  373  375  376

 116 0039H     1  K             BYTE
                                117  118  120  121  122

   2              KILLCONRSX    LITERALLY
                                  9

   2              KILLJOURNALRSX LITERALLY
                                613

   2              KILLLSTRSX    LITERALLY
                                586

 433 0DA9H   111  KILLRSX       PROCEDURE STACK=000AH
                                576  595  602  621

  35 0000H     1  LEN0          BYTE EXTERNAL(7)

  37 0000H     1  LEN1          BYTE EXTERNAL(9)

 127 0049H     1  LENFOUND      BYTE
                                198  206  209

 127 0048H     1  LENNEW        BYTE
                                205  206
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
127 0044H     1  LETTERINWORD        BYTE
                                      145  150  151  153

  2               LF                 LITERALLY
                                       11  229  393  536  538

127 0000H     1  LIST                BYTE BASED(LISTPTR) ARRAY(1)
                                      132  135  151  153

125 003AH     2  LISTPTR             ADDRESS PARAMETER
                                      126  127

  2               LISTTYPE           LITERALLY
                                      352  444  583

  2               LIT                LITERALLY

118 06ACH         LOOP               LABEL
                                      123

                  LOW                BUILTIN
                                      401  409  529

 72 0610H    16  MAKEFILE            PROCEDURE ADDRESS STACK=0002H
                                      400  426

 53 0029H     1  MAX                 BYTE PARAMETER
                                       56   57

    0000H         MEMORY             BYTE ARRAY(0)

 80 0030H     1  MODE                BYTE PARAMETER
                                       81   82

 19 0000H         MON1               PROCEDURE EXTERNAL(0) STACK=0000H
                                       44   51   58   78   82   96  107  297  309  347  447
                                      532  544

 23 0000H         MON2               PROCEDURE BYTE EXTERNAL(1) STACK=0000H
                                       40   47   66   88

 27 0000H         MON3               PROCEDURE ADDRESS EXTERNAL(2) STACK=0000H
                                       63   70   74  100  103

                  MOVE               BUILTIN
                                      505

450 006EH     1  NEGATE              BYTE
                                      452  456  463  469  475  481  487

 55 0000H     2  NEWMAX              ADDRESS BASED(BUFFERADDRESS)
                                       57   60

105 0036H     2  NEWVALUE            ADDRESS PARAMETER
                                      106  107

201 0768H         NEXTOPT            LABEL
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                212  215

  241 0053H     1  NLINES         BYTE
                                269  279

    8 0006H     1  NOCHARS        BYTE
                                539  542  545  549

  331 0B9CH        NXTCHR         LABEL
                                343

  125 003CH     2  OFFPTR         ADDRESS PARAMETER
                                126  127

   90 0032H     1  OFFSET         BYTE PARAMETER
                                 91   93

   84 0031H     1  OFFSET         BYTE PARAMETER
                                 85   86

   14 0000H     1  OFFSET         BYTE MEMBER(SCBPD)
                                 86   93

  127 0000H     1  OFFSETS        BYTE BASED(OFFPTR) ARRAY(1)
                                131  137  205

   12 000BH    87  OPTIONS        BYTE ARRAY(87) DATA
                                563  567  569  571  587  589  591  603  606  614  617

   12 0062H    15  OPTIONSOFFSET  BYTE ARRAY(15) DATA
                                563  567  569  571  587  589  591  603  606  614  617

  125 06DAH   303  OPTSCANNER     PROCEDURE STACK=0004H
                                453  457  563  567  569  571  587  589  591  603  606
                                614  617

  449 0E18H   241  OUTPUTOPTIONS PROCEDURE STACK=000AH
                                513

   15 0000H     1  OUTPUTTYPE     BYTE MEMBER(PUTPB)
                                352  355  444  566  583  609

  112 0000H     2  PARAMBLOCK     ADDRESS PARAMETER
                                113

  109 0000H        PARSE          PROCEDURE ADDRESS EXTERNAL(11) STACK=0000H
                                499

   16 0018H     4  PARSEFN        STRUCTURE
                                497  498  499  502

   34 0000H     2  PASS0          ADDRESS EXTERNAL(6)

   36 0000H     2  PASS1          ADDRESS EXTERNAL(8)

   17 001CH     8  PASSWD         BYTE ARRAY(8)
                                329  332  342  397  398  424  505
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
109 0000H     2  PFCB              ADDRESS PARAMETER
                                   110

 18 0264H        PLM               LABEL PUBLIC
                                   528

 49 05B7H    16  PRINTBUF          PROCEDURE STACK=0002H
                                   285  286  290  291  292  293  294  295  302  303  308
                                   328  351  353  354  356  357  366  390  406  439  445
                                   446  531  536  537  538

 42 059EH    16  PRINTCHAR         PROCEDURE STACK=0002H
                                   228  229  249  257  259  369  370  374  376

246 0A9CH   107  PRINTCOMMAND      PROCEDURE STACK=0006H
                                   281  283

359 0C3BH   114  PRINTFN           PROCEDURE STACK=0006H
                                   392  430  440

  2             PROC              LITERALLY
                                   227

493 0F09H   116  PROCESSFILE       PROCEDURE STACK=000CH
                                   551  573  593  619

381 0000H     1  PROGFLAG          BYTE BASED(A)
                                   387

 15 0003H     1  PROGRAMFLAG       BYTE MEMBER(PUTPB)
                                   482  483  488  489

  1 0261H   820  PUT               PROCEDURE STACK=000EH

112 0000H        PUTF              PROCEDURE EXTERNAL(12) STACK=0000H
                                   431

350 0C06H    53  PUTMSG            PROCEDURE STACK=0004H
                                   391  429  443

 12 0071H    37  PUTOPTIONS        BYTE ARRAY(37) DATA
                                   453  457

 12 0096H     7  PUTOPTIONSOFFSET  BYTE ARRAY(7) DATA
                                   453  457

 15 0014H     4  PUTPB             STRUCTURE INITIAL
                                   352  355  431  444  464  465  470  471  476  477  482
                                   483  488  489  566  583  584  609  610  611

 39 0595H     9  READCONSOLE       PROCEDURE BYTE STACK=0002H
                                   304

 53 05C7H    39  READCONSOLEBUF    PROCEDURE BYTE STACK=0002H
                                   539

241 0054H     1  REM               BYTE
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                270  283

301 005DH     1  RESPONSE         BYTE
                                304  306

329 0B83H        RETRY           LABEL
                                336  340

 80 0630H    16  RETURNERRORS    PROCEDURE STACK=0002H
                                396  419  423  428

 98 067AH    16  RSXCALL         PROCEDURE ADDRESS STACK=0002H
                                383  388  437  438

 10 0008H     1  RSXFCBPB        BYTE INITIAL
                                383  437  585  612

  9 0007H     1  RSXKILLPB       BYTE INITIAL
                                388  438  586  613

 98 0034H     2  RSXPB           ADDRESS PARAMETER
                                 99  100

231 004CH     2  S               ADDRESS PARAMETER
                                232  234  237

127 0047H     1  SAVEINDEX       BYTE
                                199  219

 14 0010H     4  SCBPD           STRUCTURE
                                 86   87   88   93   94   95   96

115 06A3H    55  SEPARATOR       PROCEDURE BYTE STACK=0000H
                                160  164  172  183  189

 14 0001H     1  SET             BYTE MEMBER(SCBPD)
                                 87   94

105 0693H    16  SETCONSOLEMODE  PROCEDURE STACK=0002H

 76 0620H    16  SETDMA          PROCEDURE STACK=0002H
                                397

 90 0658H    34  SETSCBBYTE      PROCEDURE STACK=0002H

142 085FH    30  SETUP           PROCEDURE STACK=0002H
                                180  192  202

246 005AH     1  SIZE            BYTE PARAMETER
                                247  248  253

158 08BDH    92  SKIP            PROCEDURE STACK=0002H
                                195  217

 12 00A9H     1  SPACE           BYTE DATA
                                172
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
127 0046H     1  START              BYTE
                                     130  179  182  201

494 0071H     2  STATUS             ADDRESS
                                     496  499  500  506  509

242 0055H     2  STRINGPTR          ADDRESS
                                     243  250  265  280

 38 0000H     1  TBUFF              BYTE ARRAY(1) EXTERNAL(10)
                                     534  539  541  542  543  546  548  550  551  555  558
                                     559  560

149 0B7DH    64  TESTLETTER         PROCEDURE STACK=0000H
                                     186  210

  2              TRUE               LITERALLY
                                      15  184  208  256  456  465  470  477  482  489  526
                                     610

379 0CADH   252  TRYOPEN            PROCEDURE STACK=000AH
                                     507  514

242 0057H     2  TSTRINGPTR         ADDRESS
                                     244  260  265  280

312 0B4BH    30  UCASE              PROCEDURE BYTE STACK=0000H
                                     321  546

299 0B07H    68  USERABORT          PROCEDURE STACK=0006H
                                     393  407

127 004AH     1  VALID              BYTE
                                     152  154  184  187  200  208  211

 14 0002H     2  VALUE              ADDRESS MEMBER(SCBPD)
                                      95

 90 0033H     1  VALUE              BYTE PARAMETER
                                      92   95

 62 05EEH     9  VERSION            PROCEDURE ADDRESS STACK=0002H
                                     529

 11 0000H    11  WARNING            BYTE ARRAY(11) DATA
                                     390  406

127 0042H     1  WRDPOS             BYTE
                                     131  132  135  137  145
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

MODULE INFORMATION:

```
CODE AREA SIZE     = 0FB7H   4023D
VARIABLE AREA SIZE = 0075H    117D
MAXIMUM STACK SIZE = 000EH     14D
```

```
    975 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 PUT
0000 MCDSOA
0004 BDISK    0080 BUFF      0080 BUFFA     0050 CMDRV     0000 CPU
007C CR             006D DOLLA     005C FCB       006C FCB16     005C FCBA
005C IFCB     005C IFCBA     0003 IOBYTE    0053 LEN0      0056 LEN1
0006 MAXB     0006 MEMSIZ    0005 MON1      0005 MON2      0005 MON2A
0005 MON3     0000 OFFSET    006E PARMA     0051 PASS0     0054 PASS1
007F RO             007D RR        007D RRECA     007F RRECO     005C SFCB
0080 TBUFF
0000 PUT
1699 MEMORY   1624 CCPFLAG   1625 CONWIDTH  1626 I
1627 BEGINBUFFER             1629 BUFLENGTH 162A NOCHARS   162B RSXKILLPB
162C RSXFCBPB 0180 WARNING   018B OPTIONS   01E2 OPTIONSOFFSET
01F1 PUTOPTIONS       0216 PUTOPTIONSOFFSET         021D ENDLIST   021E DELIMITERS
0229 SPACE    162D J         162E BUFPTR    1630 INDEX     1631 ENDBUF
1632 DELIMITER 1633 ENDOFSTRING            1634 SCBPD     1638 PUTPB
163C PARSEFN  1640 PASSWD    03E4 PLM       0715 READCONSOLE
071E PRINTCHAR 1648 CHAR     072E CONIN     0737 PRINTBUF
1649 BUFFERADDRESS           0747 READCONSOLEBUF
164B BUFFERADDRESS           164D MAX       076E VERSION
0777 CHECKCONSTAT            0780 DELETEFILE 164E FCBADDRESS 0790 MAKEFILE
1650 FCB      07A0 SETDMA    1652 DMA       07B0 RETURNERRORS
1654 MODE     07C0 GETSCBBYTE 1655 OFFSET   07DB SETSCBBYTE 1656 OFFSET
1657 VALUE    07FA RSXCALL   1658 RSXPB     080A GETCONSOLEMODE
0813 SETCONSOLEMODE          165A NEWVALUE  0823 SEPARATOR 165C CHARACTER
165D K        082C LOOP      085A OPTSCANNER 165E LISTPTR   1660 OFFPTR
1662 IDXPTR   1664 I         1665 J         1666 WRDPOS    1667 CHARACTER
1668 LETTERINWORD            1669 FOUNDFIRST 166A START     166B SAVEINDEX
166C LENNEW   166D LENFOUND  166E VALID     0989 CHECKINLIST
166F I        09DF SETUP     09FD TESTLETTER 0A3D SKIP      0A99 EATBLANKS
08C3 EXIT1    08D3 GOOD      08E8 NEXTOPT   0959 FINISHED  0AB5 CRLF
0AC0 FILL     1670 S         1672 F         1673 C         0AEB ERROR
1674 CODE     1675 I         1676 J         1677 NLINES    1678 REM
1679 STRINGPTR 167B TSTRINGPTR 167D CAROTFLAG 0C1C PRINTCOMMAND
167E SIZE     0C87 USERABORT 167F A         1681 RESPONSE  0CCB UCASE
1682 CHAR     0CE9 GETUCASE  1683 C         0CF7 GETPASSWD 1684 I
1685 C        0D03 RETRY     0D1C NXTCHR    0D86 PUTMSG    0DBB PRINTFN
1686 FCBAD    1688 K         0E2D TRYOPEN   1689 ERRORCODE 168B A
168D CODE     0F29 KILLRSX   168E FCBADR     1690 A
0F98 OUTPUTOPTIONS           1692 NEGATE    1089 PROCESSFILE
1693 BUFADR   1695 STATUS    10FD INPUTFOUND 1697 BUFFERADR
0000 PARSE
12B0 DBLNK1   1274 DEBLNK    1232 DELIM     126D DELIM1    1270 DELIM2
122A PADFLD   11C8 PARS81    11C9 PARS82    1137 PARSE     1158 PARSE1
1186 PARSE2   1189 PARSE3    119B PARSE4    119F PARSE5    11A1 PARSE6
11B5 PARSE7   11B9 PARSE8    11DA PARSE9    1203 PWFLD     1216 PWFLD1
11F5 SETFD1   11F7 SETFD2    11FB SETFD3    11E0 SETFLD
0000 MODULE
FFFF BIOSFU   007F BIOSGN    00B3 CCPFLG    0001 CINF      0006 CINJMP
0010 CLOSEF   00F9 COMMON    0074 CONINF    00CF CONMOD    007A CONOUF
006E CONSTF   0009 COUJMP    0002 COUTF     000D CR        0006 CRAWF
000A CREADF   000B CSTATF    0003 CSTJMP    0019 CURDRV    0013 DELF
001A DMAF     001F DPBF      0014 DREADF    00FF EOT       00AA ERRFLG
0000 EVERYT   0000 FALSE     0082 GFCBF     0080 GINITF    0081 GKILLF
008E JFCBF    008C JINITF    008D JKILLF    00C3 JMPI      0070 LBLKF
0005 LCHRF    000A LF        00D4 LISTCP    0080 LISTFX    000C LSTJMP
0021 LXIH     0016 MAKEF     00FF NOREST    000F OPENF     006F PBLKF
0009 PBUFF    0086 PCFCBF    0085 PCKILL    0084 PINITF    008A PLFCBF
0089 PLKILL   0063 RESALV    0025 RESDVF    000D RESETF    003C RSXF
003A SCBA     0031 SCBF      000E SELECT    0091 SFCBF     0090 SKILLF
00FE SKIPF    0040 SUBMIT    00A0 SVKILL    FFFF TRUE      0020 USERF
0000 WBOOT    0001 WBOOTA    0068 WBOOTF    1525 AUX       1537 BDOSFU
1406 BIOSCK   140E BIOSER    152E BIOSFI    152C BIOSJM    1538 BIOSMO
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

1523 BIOSOF	152A BIOSOU	154D BLM	1571 CBUFP	1364 CHECK
1318 CKINPU	1549 COOKED	142F DATA49	1548 ECHO	13EE ERROR
1387 FIX0	1397 FIX1	13AB FIX3	1546 FUNCEN	1539 FUNCTB
1348 GETRSX	1427 GETSCB	141F GETUSR	1528 GOBIOS	1526 INITTA
1546 INPUT	1449 INPUTF	1431 LDIR	1547 LIST	143C LISTFU
14B7 MEMERR	1572 [illegible]END	12C4 MOVFCB	1526 MOVSTA	1458 NOBIOS
14D6 NOGET	147F NOTDON	13EB NOTIMP	154B PFCBF	154A PKILLF
154E PROGRA	1570 PUTCR	1284 PUTF	1550 PUTFCB	155E PUTMOD
155F PUTRC	154F PUTUSR	154C RECORD	1522 RESOFF	154A RSXFUN
1457 RSXINI	13DC RSXMOV	1456 SAVKIL	1526 SCBADR	1295 SETLST
1421 SETUSR	13AD TRAP	1535 WMFIX	1533 WMJMP	1531 WMSTA

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 PUT#
0000 MCD80A#
0000 PUT#
0715 39         0715 40         071E 41         071E 42         0722 44
072D 45         072E 46         072E 47         0737 48         0737 49
073D 51         0746 52         0747 53         074F 57         0759 58
0762 59         0769 60         076E 61         076E 62         076E 63
0777 64         0777 65         0777 66         0780 67         0780 68
0786 70         0790 71         0790 72         0796 74         07A0 75
07A0 76         07A6 78         07AF 79         07B0 80         07B4 82
07BF 83         07C0 84         07C4 86         07CA 87         07CF 88
07D8 89         07D8 90         07DE 93         07E4 94         07E9 95
07F1 96         07F9 97         07FA 98         0800 100        080A 101
080A 102        080A 103        0813 104        0813 105        0819 107
0822 108        0823 115        0827 117        082C 118        083B 119
083E 120        084E 121        0852 122        0856 123        0859 124
085A 125        0989 128        0989 130        098F 131        099A 132
09AB 133        09AF 134        09B6 135        09C7 136        09C8 137
09D6 138        09D9 139        09DE 140        09DF 141        09DF 142
09DF 143        09E6 144        09E9 145        09EF 146        09F4 147
09FC 148        09FD 149        09FD 150        0A01 151        0A0F 152
0A17 153        0A2B 154        0A2D 155        0A31 156        0A3C 157
0A3D 158        0A3D 159        0A4B 160        0A55 161        0A6A 162
0A6E 163        0A79 164        0A83 165        0A86 166        0A8C 167
0A98 168        0A99 169        0A99 170        0A99 172        0AAA 173
0AB1 174        0AB4 175        086B 176        0873 177        0874 178
0877 179        087C 180        087F 181        0888 182        088F 183
0899 184        089E 185        08A6 186        08A9 187        08B0 188
08B3 189        08BD 190        08C0 191        08C3 192        08C6 193
08C9 194        08CF 195        08D2 196        08D3 197        08D9 198
08DC 199        08E3 200        08E8 201        08EF 202        08F2 203
08FB 204        08FE 205        091A 206        0922 208        0927 209
0936 210        0939 211        0940 212        0943 213        094A 214
094D 215        0950 216        0955 217        0958 218        0959 219
0960 220        096E 221        0971 222        0979 223        0983 224
0988 225        0989 226        0AB5 227        0AB5 228        0ABA 229
0ABF 230        0AC0 231        0ACD 235        0AD9 236        0AE0 237
0AE7 238        0AEA 239        0AEB 240        0C1C 246        0C20 248
0C2F 249        0C36 250        0C3D 251        0C44 252        0C47 253
0C56 254        0C63 256        0C68 257        0C6D 258        0C70 259
0C75 260        0C7C 261        0C83 262        0C86 263        0AEF 264
0AF4 265        0AFD 266        0B05 267        0B0D 268        0B12 269
0B24 270        0B32 271        0B3A 273        0B52 274        0B5C 275
0B64 276        0B73 277        0B73 278        0B76 279        0B85 280
0B8B 281        0B92 282        0B99 283        0BA0 284        0BA7 285
0BB0 286        0BB6 287        0BBE 288        0BC1 289        0BD1 290
0BDA 291        0BE3 292        0BEC 293        0BF5 294        0BFE 295
0C07 296        0C13 297        0C1B 298        0C87 299        0C8D 302
0C95 303        0C9B 304        0CA1 305        0CA4 306        0CBC 308
0CC2 309        0CCA 310        0CCA 311        0CCB 312        0CCF 314
0CD7 315        0CDF 316        0CE5 317        0CE9 318        0CE9 319
0CE9 321        0CF3 322        0CF7 323        0CF7 324        0CF7 326
0CFA 327        0CFD 328        0D03 329        0D0E 330        0D1C 331
0D27 332        0D34 333        0D3C 334        0D3D 335        0D45 336
0D48 337        0D50 339        0D58 340        0D5B 342        0D6B 343
0D6E 344        0D6E 345        0D6E 346        0D76 347        0D7E 348
0D85 349        0D86 350        0D86 351        0D8C 352        0D94 353
0D9D 354        0DA3 355        0DAB 356        0DB4 357        0DBA 358
0DBB 359        0DC1 364        0DCB 365        0DCE 366        0DD4 367
0DDD 369        0DE7 370        0DEC 371        0DEC 372        0DFA 373
0E02 374        0E07 375        0E17 376        0E25 377        0E2C 378
0E2D 379        0E2D 383        0E36 384        0E42 386        0E4A 387
0E4F 388        0E5B 390        0E61 391        0E64 392        0E6C 393
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0E72 394    0E72 395    0E72 396    0E77 397    0E7D 398
0E85 399    0E8D 400    0E96 401    0E9F 403    0EA6 404
0EAB 406    0EB1 407    0EB7 408    0EC0 409    0EC9 411
0ED0 412    0ED5 413    0EDA 414    0EE2 416    0EE5 417
0EE8 418    0EE8 419    0EED 420    0EF6 421    0EF6 422
0EF6 423    0EFB 424    0F03 425    0F0B 426    0F14 427
0F14 428    0F19 429    0F1C 430    0F22 431    0F28 432
0F29 433    0F29 435    0F41 436    0F46 437    0F58 438
0F61 439    0F67 440    0F6F 441    0F72 442    0F75 443
0F78 444    0F80 445    0F89 446    0F8F 447    0F97 448
0F98 449    0F98 451    0FB0 452    0FB5 453    0FC2 454
0FCA 456    0FCF 457    0FDC 458    0FDC 459    0FF4 460
0FF9 461    1001 463    1008 464    1010 465    1015 466
1015 467    101D 469    1024 470    102C 471    1031 472
1031 473    1039 475    1040 476    1048 477    104D 478
104D 479    1055 481    105C 482    1064 483    1069 484
1069 485    1071 487    1078 488    1080 489    1085 490
1085 491    1088 492    1089 493    108F 497    1095 498
109B 499    10A4 500    10B1 502    10B7 503    10BC 504
10BC 505    10CC 506    10D8 507    10DE 509    10E5 510
10EE 511    10F6 513    10F9 514    10FC 515    10FC 516
10FC 517    10FD 518    1103 521    111D 522    1124 523
1127 524    1130 525    1133 526    1136 527    03E1 528
03E7 529    03FF 531    0405 532    040D 533    040D 534
0418 536    041E 537    0424 538    042A 539    0435 540
0438 541    043D 542    0448 543    0453 544    045B 545
046A 546    0481 547    048B 548    0491 549    0497 550
049A 551    04A0 552    04A3 554    04A8 555    04B7 556
04BE 557    04C1 558    04C7 559    04CC 560    04D7 561
04DC 562    04E1 563    04EE 564    0512 566    0517 567
0524 568    052C 569    0539 570    0541 571    054E 572
0556 573    0561 575    0585 576    058B 577    0590 578
0590 579    0593 581    05B7 583    05BC 584    05BF 585
05C4 586    05C7 587    05D4 588    05DC 589    05E9 590
05F1 591    05FE 592    0606 593    060E 594    0632 595
0638 596    063D 597    0640 599    0664 601    066C 602
0672 603    067F 604    067F 605    0687 606    0697 607
069F 609    06A4 610    06A7 611    06AA 612    06AF 613
06B2 614    06BF 615    06BF 616    06C7 617    06D4 618
06DC 619    06E4 620    0708 621    070E 622    0713 623
0713 624    0713 625    0713 627
0000 PARSE#

0000 MODULE#
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                title  'PUT.RSX 3.0 - CP/M 3.0 Output Redirection - August 19
                ;*****************************************************************
                ;
                ;       PUT  'Output Redirection Facility'  version 3.0
                ;
                ;       11/30/82 - Doug Huskey
                ;       This RSX redirects console or list output to a file.
                ;*****************************************************************
                ;
                ;
                ;       generation procedure
                ;
                ;       rmac putrsx
                ;       xref putrsx
                ;       link putrsx[op]
                ;       ERA put.RSX
                ;       REN put.RSX=putRSX.PRL
                ;       GENCOM put.com put.rsx
                ;
                ;       initialization procedure
                ;
                ;       PUTF makes a RSX function 60 call with a sub-function of
                ;       128.  PUTRSX returns the address of a data table containing:
                ;
                ;       init$table:
                ;               dw      kill            ;remove PUT at warmboot flg
                ;               dw      0               ;reserved
                ;               dw      bios$output     ;BIOS entry point into PUT
                ;               dw      putfcb          ;FCB address
                ;
                ;       PUTF initializes the data are between movstart: and movend:
                ;       and moves it into PUT.RSX.  This means that data should not
                ;       be reordered without also changing PUTF.ASM.
                ;
                ;
FFFF =          true            equ     0ffffh
0000 =          false           equ     0000h
                ;
FFFF =          bios$functions equ      true    ;intercept BIOS console functions
0000 =          remove$rsx      equ     false   ;this RSX does its own removal
                ;
                ;       low memory locations
                ;
0000 =          wboot   equ     0000h
0001 =          wboota equ      wboot+1
0005 =          bdos    equ     0005h
0006 =          bdosl   equ     bdos+1
0080 =          buf     equ     0080h
                ;
                ;       equates for non graphic characters
                ;
0003 =          ctlc    equ     03h     ; control c
0005 =          ctle    equ     05h     ; physical eol
0008 =          ctlh    equ     08h     ; backspace
0010 =          ctlp    equ     10h     ; prnt toggle
0012 =          ctlr    equ     12h     ; repeat line
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
 57  0013 =      ctls   equ    13h     ; stop/start screen
 58  0015 =      ctlu   equ    15h     ; line delete
 59  0018 =      ctlx   equ    18h     ; =ctl-u
 60  001A =      ctlz   equ    1ah     ; end of file
 61  007F =      rubout equ    7fh     ; char delete
 62  0009 =      tab    equ    09h     ; tab char
 63  000D =      cr     equ    0dh     ; carriage return
 64  000A =      lf     equ    0ah     ; line feed
 65  005E =      ctl    equ    5eh     ; up arrow
 66              ;
 67              ;      BDOS function equates
 68              ;
 69  0001 =      cinf   equ    1       ;read character
 70  0002 =      coutf  equ    2       ;output character
 71  0006 =      crawf  equ    6       ;raw console I/O
 72  000A =      creadf equ    10      ;read buffer
 73  000B =      cstatf equ    11      ;status
 74  0005 =      lchrf  equ    5       ;print character
 75  0009 =      pbuff  equ    9       ;print buffer
 76  000D =      resetf equ    13      ;reset drive
 77  000F =      openf  equ    15      ;open file
 78  0010 =      closef equ    16      ;close file
 79  0013 =      delf   equ    19      ;delete file
 80  0014 =      dreadf equ    20      ;disk read
 81  0015 =      writef equ    21      ;disk write
 82  001A =      dmaf   equ    26      ;set dma function
 83  0020 =      userf  equ    32      ;set/PUT user number
 84  0025 =      resdvf equ    37      ;reset drive function
 85  0030 =      flushf equ    48      ;flush buffers function
 86  0031 =      scbf   equ    49      ;set/PUT system control block word
 87  003B =      loadf  equ    59      ;Program load function
 88  003C =      rsxf   equ    60      ;RSX function call
 89  0062 =      resalvf       equ     98      ;reset allocation vector
 90  006F =      pblkf  equ    111     ;print block to console
 91  0070 =      lblkf  equ    112     ;print block to list device
 92  0080 =      ginitf equ    128     ;GET initialization sub-function no.
 93  0081 =      gkillf equ    129     ;GET delete sub-function no.
 94  0082 =      gfcbf  equ    130     ;GET file display sub-function no.
 95  0084 =      pinitf equ    132     ;PUT initialization sub-function no.
 96  0085 =      pckillf       equ     133     ;PUT console delete sub-function no.
 97  0089 =      plkillf       equ     137     ;PUT list delete sub-function no.
 98  0086 =      pcfcbf equ    134     ;return PUT console fcb address
 99  008A =      plfcbf equ    138     ;return PUT list fcb address
100  008C =      jinitf equ    140     ;JOURNAL initialization sub-function no.
101  008D =      jkillf equ    141     ;JOURNAL delete sub-function no.
102  008E =      jfcbf  equ    142     ;return JOURNAL fcb address
103              ;
104              ;      System Control Block definitions
105              ;
106  003A =      scba   equ    03ah    ;offset of scbadr from SCB base
107  00B3 =      ccpflg equ    0b3h    ;offset of ccpflags word from page boundary
108  0020 =      ccpres equ    020h    ;ccp resident flag = bit 5
109  00FE =      bdosoff equ   0feh    ;offset of BDOS address from page boundary
110  00AA =      errflg equ    0aah    ;offset of error flag from page boundary
111  00CF =      conmode       equ     0cfh    ;offset of console mode word from pag. bound.
112  00D3 =      outdel equ    0d3h    ;offset of print buffer delimiter
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
00D4 =          listcp equ      0d4h    ;offset of ^P flag from page boundary
00E0 =          usrcode         equ     0e0h    ;offset of user number from pg bnd.
00E1 =          dcnt   equ      0e1h    ;offset of dcnt, searcha & searchl from pg bnd.
006E =          constfx         equ     06eh    ;offset of constat JMP from page boundary
0074 =          coninfx         equ     074h    ;offset of conin JMP from page boundary
                ;
                ;
                ;*****************************************************************
                ;               RSX HEADER
                ;*****************************************************************

0000 0000000000serial:          db      0,0,0,0,0,0

                trapjmp:
0006 C33400             jmp     trap            ;trap read buff and DMA functions
0009 C30000     next:   jmp     0               ;go to BDOS
000C 0500       prev:   dw      bdos
000E FF         kill:   db      0FFh            ;Remove at wstart if not zero
000F 00         nbank:  db      0
0010 5055542020rname:   db      'PUT     '      ;RSX name
0018 0000       space:  dw      0
001A 00         patch:  db      0

                ;*****************************************************************
                ;               START OF CODE
                ;*****************************************************************
                ;
                ;       ABORT ROUTINE
                ;
                puteof:                                 ;close output file and abort
001B 3A1003             lda     cbufp
001E B7                 ora     a
001F CA9C01             jz      restor
0022 1E1A               mvi     e,ctlz
0024 CD3401             call    putc
0027 C31B00             jmp     puteof


                ;
                ;*****************************************************************
                ;               BIOS TRAP ENTRY POINT
                ;*****************************************************************
                ;
                ;
                ;       ARRIVE HERE ON EACH INTERCEPTED BIOS CALL
                ;
                ;
                bios$output:
                        ;
                if bios$functions
                        ;
                        ;enter here from BIOS constat
002A 59                 mov     e,c             ;character in E
002B 3AD602             lda     bdosfunc        ;BDOS function to use
002E 4F                 mov     c,a
002F 3E01               mvi     a,1             ;offset in exit table = 1
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        jmp     bios$trap
                endif
                ;
                ;
                ;*****************************************************************
                ;               BDOS TRAP ENTRY POINT
                ;*****************************************************************
                ;
                ;
                ;       ARRIVE HERE AT EACH BDOS CALL
                ;
                trap:
                        ;
                if bios$functions
                        ;
0034 AF                 xra     a
                biostrap:
                        ;enter here on BIOS calls
0035 32AC02             sta     exit$off
                endif
0038 E1                 pop     h               ;return address
0039 E5                 push    h               ;back to stack
003A 3A0800             lda     trapjmp+2       ;PUT.RSX page address
003D BC                 cmp     h               ;high byte of return address
003E DA5B00             jc      exit            ;skip calls on bdos above here
0041 79                 mov     a,c
0042 FE3C               cpi     rsxf
0044 CA1502             jz      rsxfunc         ;check for initialize or abort
0047 FE1A               cpi     dmaf
0049 CA0402             jz      dmafunc         ;save users DMA address
004C FE0E               cpi     14              ;reset function + 1
004E DA6A00             jc      tbl$srch        ;search if func < 14
0051 FE62               cpi     98
0053 D26A00             jnc     tbl$srch        ;search if func >= 98
0056 FE25               cpi     resdvf
0058 CA6A00             jz      tbl$srch        ;search if func = 37
                        ;
                        ;       EXIT - FUNCTION NOT MATCHED
                        ;
                exit:

                if not bios$functions
                        ;
                exit1:  jmp     next            ;go to next RSX or BDOS

                else
005B 3AAC02             lda     exit$off        ;PUT type of call:
005E 219103     exit1:  lxi     h,exit$table    ;0=BDOS call, 1=BIOS call
                endif

                tbl$jmp:

                        ;  a = offset (rel 0)
                        ; hl = table address
0061 87                 add     a               ;double for 2 byte addresses
0062 CDF601             call    addhla          ;HL = .(exit routine)
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
225  0065 46          mov   b,m          ;get low byte from table
226  0066 23          inx   h
227  0067 66          mov   h,m
228  0068 68          mov   l,b          ;HL = exit routine
229  0069 E9          pchl               ;gone to BDOS or BIOS
230
231              tbl$srch:
232
233                    ;
234                    ;CHECK IF THIS FUNCTION IS IN FUNCTION TABLE
235                    ;if matched b = offset in table (rel 0)
236                    ;FF terminates table
237                    ;FE is used to mark non-intercepted functions
238                    ;
239  006A 21D802       lxi   h,func$tbl   ;list of intercepted functions
240  006D 0600         mvi   b,0          ;start at beginning
241              tbl$srch1:
242  006F 7E          mov   a,m          ;get next table entry
243  0070 B9          cmp   c            ;is it the same?
244  0071 CA7D00      jz    intercept    ;we found a match, B = offset
245  0074 04          inr   b
246  0075 23          inx   h
247  0076 3C          inr   a            ;0FFh terminates list
248  0077 C26F00      jnz   tbl$srch1    ;try next one
249  007A C35B00      jmp   exit         ;end of table - not found
250
251              ;
252              ;
253              ;*****************************************************************
254              ;             REDIRECTION PROCESSOR
255              ;*****************************************************************
256              ;
257              ;
258              ;     INTERCEPTED BDOS FUNCTIONS ARRIVE HERE
259              ;
260              ;     enter with
261              ;                    B = routine offset in table
262              ;                    C = function number
263              ;                   DE = BDOS parameters
264
265              intercept:
266
267                    ;switch to local stack
268  007D 210000      lxi   h,0
269  0080 39          dad   sp
270  0081 223104      shld  oldstack
271  0084 313104      lxi   sp,stack
272
273              redirect:
274
275  0087 D5          push  d            ;save info
276  0088 C5          push  b            ;save function
277  0089 2AC502      lhld  scbadr
278                    ;
279                    ;are we active now?
280                    ;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        lda     program
008F B7                 ora     a               ;program output only?
0090 C4FB01             cnz     ckccp           ;if not, test if CCP is calling
0093 CAA600             jz      cklist          ;jump if not CCP or program output
0096 79                 mov     a,c
0097 FE0A               cpi     0ah             ;is it function 10?
0099 C21001             jnz     skip            ;skip if not
009C 219903             lxi     h,ccpcnt        ;decrement once for each
009F 35                 dcr     m               ;CCP function 10
00A0 FC1B00             cm      puteof          ;if 2nd appearance of CCP
00A3 C31001             jmp     skip            ;if CCP is active
                        ;
                        ;check for list processing and ^P status
                        ;
                cklist:
00A6 3AE602             lda     list
00A9 B7                 ora     a               ;list redirection?
00AA CABA00             jz      ckecho          ;jump if not
00AD 2ED4               mvi     l,listcp        ;HL = .^P flag
00AF 7E                 mov     a,m
00B0 B7                 ora     a               ; ^P on?
00B1 C2BD00             jnz     setecho         ;set echo on if so
00B4 78                 mov     a,b
00B5 FE02               cpi     2               ;console function?
00B7 D21001             jnc     skip            ;skip if so
00BA 3AE702     ckecho: lda     echoflg         ;echo parameter
                setecho:
00BD 329803             sta     echo
                        ;
                        ;go to function trap routine
                        ;
                gofunct:
00C0 210901             lxi     h,retmon        ;program return routine
00C3 E5                 push    h               ;push on stack
00C4 78                 mov     a,b             ;offset
00C5 21AD02             lxi     h,trap$tbl
00C8 C36100             jmp     tbl$jmp         ;go to table address
                ;
                ;
                rawio:
                        ;direct console i/o - read if 0ffh
                        ;returns to retmon
00CB 7B                 mov     a,e
00CC FEFD               cpi     0fdh
00CE DA1901             jc      putchr
00D1 FEFE               cpi     0feh
00D3 C8                 rz                      ;make the status call  (FE)
00D4 DAE100             jc      conin           ;make the input call   (FD)
00D7 CD0900             call    next            ;call for input/status (FF)
00DA B7                 ora     a
00DB CA0301             jz      retmon1
00DE C3E400             jmp     conin1
                        ;
                        ;input function
                        ;
                conin:
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
00E1 CD5B00          call    exit            ;make the call
00E4 5F        conin1:       mov     e,a             ;put character in E
00E5 F5              push    psw             ;save character
00E6 CD1E01          call    conout          ;put character into file
00E9 F1              pop     psw             ;character in A
                     ;
                     ;       RETURN FROM FUNCTION TRAP ROUTINE
                     ;
00EA FE0D            cpi     cr
00EC C20301          jnz     retmon1

                retmon2:
                     ;output linefeed before returning
00EF F5              push    psw             ;save character
00F0 3A9803          lda     echo
00F3 B7              ora     a               ;no echo mode
00F4 1E0A            mvi     e,lf
00F6 0E02            mvi     c,coutf
00F8 CC0900          cz      next            ;output lf if so
00FB 3AE502          lda     input
00FE B7              ora     a
00FF C41E01          cnz     conout
0102 F1              pop     psw             ;restore character

                retmon1:
                     ;return to calling program
0103 2A3104          lhld    old$stack
0106 F9              sphl
0107 6F              mov     l,a
                retmon0:
0108 C9              ret                     ;to calling program
                     ;
                retmon:
                     ;echo before returning?
0109 3A9803          lda     echo
010C B7              ora     a
010D CA0301          jz      retmon1         ;return to program if no echo
                     ;otherwise continue
                     ;
                     ;       PERFORM INTERCEPTED BDOS CALL
                     ;
                skip:
                     ;restore BDOS call and stack
0110 C1              pop     b               ;restore BDOS function no.
0111 D1              pop     d               ;restore BDOS parameter
0112 2A3104          lhld    old$stack
0115 F9              sphl
0116 C35B00          jmp     exit            ;goto BDOS

                ;*****************************************************************
                ;               BIOS FUNCTIONS (REDIRECTION ROUTINES)
                ;*****************************************************************
                ;
                putchr:
                     ;put out character in E unless putting input
0119 3AE502B7C0      lda input! ora a! rnz   ;return (retmon) if input redirection
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        listf:
                        conout:
                        conoutf:
                        ctlout:
                                ;send E character with possible preceding up-arrow
011E 7BFE1ACA2A                 mov a,e! cpi ctlz! jz ctlout1   ;always convert ^Z
0124 CD2F02                     call echoc      ;cy if not graphic (or special case)
0127 D23401                     jnc putc        ;skip if graphic, tab, cr, lf, or ctlh

                                ctlout1:
                                        ;send preceding up arrow
012A F51E5ECD34                         push psw! mvi e,ctl! call putc ;up arrow
0130 F1F640                             pop psw! ori 40h ;becomes graphic letter
0133 5F                                 mov e,a ;ready to print
                                        ;(drop through to PUTC)
                        ;
                        ;
                        ;       put next character into file
                        ;
                        ;
                        putc:   ;write sector if full, close in each physical block
                                ;abort PUT if any disk error occurs
                                ;character in E
0134 211003                     lxi     h,cbufp
0137 7E                         mov     a,m             ; A = cbufp
0138 E5                         push    h
0139 23                         inx     h               ;HL = .cbuf
013A CDF601                     call    addhla          ;HL = .char
013D 73                         mov     m,e             ;store character
013E E1                         pop     h
013F 34                         inr     m               ;next chr position
0140 F0                         rp                      ;minus flag set after 128 chars
                        ;
                        ;       WRITE NEXT RECORD
                        ;
                        write:
0141 0E15                       mvi     c,writef
0143 CD6101                     call    putdos
0146 C49C01                     cnz     restor          ;abort RSX if error
0149 AF                         xra     a
014A 321003                     sta     cbufp           ;reset buffer position to 0
014D 21EB02                     lxi     h,record
0150 35                         dcr     m               ;did we cross the block boundary?
0151 F0                         rp                      ;return if not
0152 CD5F01                     call    close           ;close the file if so
0155 C49C01                     cnz     restor          ;abort RSX if error
0158 21EC02                     lxi     h,blm           ;HL = .blm
015B 7E                         mov     a,m
015C 2B                         dcx     h
015D 77                         mov     m,a             ;set record = blm
015E C9                         ret
                        ;
                        ;       CLOSE THE FILE
                        ;
                        close:
015F 0E10                       mvi     c,closef
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        ;
                        ;       PUT FILE OPERATION
                        ;
                        putdos:
0161 C5                         push    b               ;function no. in C
0162 111103                     lxi     d,cbuf
0165 CDE101                     call    setdma          ;set DMA to our buffer
0168 C1                         pop     b               ;function no. in C
0169 2AC502                     lhld    scbadr
016C E5                         push    h               ;save for restore
016D 119A03                     lxi     d,sav$area      ;10 byte save area
0170 D5                         push    d               ;save for restore
0171 CDEB01                     call    mov7            ;save hash info in save area
0174 2EE0                       mvi     l,usrcode       ;HL = .BDOS user number in SCB
0176 CDEB01                     call    mov7            ;save user, dcnt, search addr, len &
0179 2B                         dcx     h               ; multi-sector count
017A 3601                       mvi     m,1             ;set multi-sector count=1
017C 2EE0                       mvi     l,usrcode       ;HL = .BDOS user number
017E 11EE02                     lxi     d,putusr
0181 1A                         ldax    d
0182 77                         mov     m,a             ;set BDOS user = putusr
0183 13                         inx     d               ;DE = .putfcb
0184 CD0900                     call    next            ;write next record or close file
0187 E1                         pop     h               ;HL = .sav$area
0188 D1                         pop     d               ;DE = .scb
0189 F5                         push    psw             ;save A (non-zero if error)
018A CDEB01                     call    mov7            ;restore hash info
018D 1EE0                       mvi     e,usrcode       ;DE = .user num in scb
018F CDEB01                     call    mov7            ;restore dcnt search addr & len
0192 2A9503                     lhld    udma
0195 EB                         xchg
0196 CDE101                     call    setdma          ;restore DMA to program's buffer
0199 F1                         pop     psw
019A B7                         ora     a
019B C9                         ret                     ;zero flag set if successful
                        ;
                        ;       CLOSE FILE AND TERMINATE RSX
                        ;
                        restor:
019C CD5F01                     call    close
019F 11A803                     lxi     d,close$err
01A2 C4E601                     cnz     msg             ;print message if close error
01A5 21FFFF                     lxi     h,0ffffh
01A8 22E902                     shld    rsxfunctions    ;set killf and fcbf to inactive
                                ;
                                ;set RSX aborted flag
                                ;
01AB 210E00                     lxi     h,kill          ;0=active, 0ffh=aborted
01AE 36FF                       mvi     m,0ffh          ;set to 0ffh (in-active)
                                ;are we the bottom RSX, if so remove ourselves immediately
                                ;to save memory
01B0 3A0700                     lda     bdosl+1         ;get high byte of top of tpa
01B3 BC                         CMP     H               ;Does location 6 point to us

                        if remove$rsx
                                jnz     bios$fixup      ;done, if not
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                        lhld    next+1
                        shld    bdosl
                        xchg
                        lhld    scbadr
                        mvi     l,bdosoff       ;HL = 'BDOS' address in SCB
                        mov     m,e             ;put next address into SCB
                        inx     h
                        mov     m,d
                        xchg
                        mvi     l,0ch           ;HL = .previous RSX field in next RSX
                        mvi     m,7
                        inx     h
                        mvi     m,0             ;put previous into previous
                    else
01B4 0E3B               mvi     c,loadf
01B6 110000             lxi     d,0
01B9 CC0900             cz      next            ;fixup RSX chain, if this RSX on bottom
                    endif

                    if bios$functions

                    bios$fixup:
                            ;
                            ;restore bios jumps
01BC 3AD702                 lda     restore$mode        ;may be FF, 7f or 0
01BF 3C                     inr     a
01C0 C8                     rz                          ; FF = no bios interception
01C1 2AD002                 lhld    wmsta               ;real warm start routine
01C4 EB                     xchg
01C5 2AD202                 lhld    wm$jmp              ;wboot jump in bios
01C8 73                     mov     m,e
01C9 23                     inx     h
01CA 72                     mov     m,d                 ;restore real routine in jump
01CB 2AC902                 lhld    biosout             ;conin,conout or list jmp
01CE EB                     xchg
01CF 2ACB02                 lhld    bios$jmp            ;address of real bios routine
01D2 73                     mov     m,e
01D3 23                     inx     h
01D4 72                     mov     m,d
01D5 F8                     rm                          ; 7f = RESBDOS jmps not changed
01D6 2AD402                 lhld    wmfix
01D9 36C3                   mvi     m,jmp               ;replace jmp for warm start
01DB 2ACD02                 lhld    biosfix
01DE 36C3                   mvi     m,jmp               ;replace jmp for other trapped jump
                    endif
01E0 C9                     ret                         ; 0  = everything done
                    ;
                    ;       set DMA address in DE
                    ;
01E1 0E1A           setdma: mvi     c,dmaf
01E3 C30900                 jmp     next
                    ;
                    ;       print message to console
                    ;
01E6 0E09           msg:    mvi     c,pbuff
01EB C30900                 jmp     next
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        ;
                        ;       move routine
                        ;
01EB 0607       mov7:   mvi     b,7
                        ;                       HL = source
                        ;                       DE = destination
                        ;                       B = count
01ED 7E         move:   mov     a,m
01EE 12                 stax    d
01EF 23                 inx     h
01F0 13                 inx     d
01F1 05                 dcr     b
01F2 C2ED01             jnz     move
01F5 C9                 ret
                        ;
                        ;       add a to hl
                        ;
01F6 85         addhla: add     l
01F7 6F                 mov     l,a
01F8 D0                 rnc
01F9 24                 inr     h
01FA C9                 ret

                        ;
                        ;       check if CCP is calling
                        ;
                        ckccp:
                                ;returns zero flag set if not CCP
01FB 2AC502             lhld    scbadr
01FE 2EB4               mvi     l,ccpflg+1      ;HL = .ccp flag 2
0200 7E                 mov     a,m
0201 E620               ani     ccpres          ;is it the CCP?
0203 C9                 ret
                        ;
                        ;*****************************************************************
                        ;               BDOS FUNCTION HANDLERS
                        ;*****************************************************************
                        ;
                        ;
                        ;       FUNCTION 26 - SET DMA ADDRESS
                        ;
                        dmafunc:
0204 EB                 xchg                    ;dma to hl
0205 229503             shld    udma            ;save it
0208 EB                 xchg
0209 C30900             jmp     next
                        ;
                        ;
                        ;       BIOS WARM START TRAP FUNCTION
                        ;
                        warmtrap:
020C 313104             lxi     sp,stack
020F CD5F01             call    close           ;close if wboot originated below RSX
0212 C3CF02             jmp     wstart
                        ;
                        ;       BDOS FUNCTION 60 - RSX FUNCTION CALL
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;
                        rsxfunc:                    ;check for initialize or delete RSX functions
0215 1A                         ldax    d           ;get sub-function number
0216 FE84                       cpi     pinitf      ;is it a PUT initialization
0218 21C502                     lxi     h,init$table
021B C8                         rz                  ;return to caller if init call
                                ;check for FCB display functions
021C 47                         mov     b,a
021D 3AEA02                     lda     fcbf        ;is it a a PUT fcb request
0220 B8                         cmp     b
0221 21EF02                     lxi     h,putfcb
0224 C8                         rz                  ;return if so
                                ;check for kill function
0225 3AE902                     lda     killf       ;local Kill (Kill only this one)
0228 B8                         cmp     b
0229 CA1B00                     jz      puteof      ;Kill and return to caller
022C C35B00                     jmp     exit        ;abort any higher PUTs

                        ;
                        ;
                        ;*****************************************************************
                        ;               BDOS OUTPUT ROUTINES
                        ;*****************************************************************
                        ;
                        ;
                        ;       July 1982
                        ;
                        ;
                        ;       Console handlers
                        ;
                        echoc:
                                ;are we in cooked or raw mode?
022F 3AE802B77B                 lda cooked! ora a! mov a,e! rz ;return if raw
                                ;echo character if graphic
                                ;cr, lf, tab, or backspace
0235 FE0DC8                     cpi cr! rz ;carriage return?
0238 FE0AC8                     cpi lf! rz ;line feed?
023B FE09C8                     cpi tab! rz ;tab?
023E FE08C8                     cpi ctlh! rz ;backspace?
0241 FE20C9                     cpi ' '! ret ;carry set if not graphic
                        ;
                        ;
                        print:
                                ;print message until M(DE) = '$'
0244 2AC502                     lhld scbadr
0247 2ED3                       mvi l,OUTDEL
0249 1ABEC8                     ldax d! CMP M! rz ;stop on delimiter
                                        ;more to print
024C 13D55F                             inx d! push d! mov e,a ;char to E
024F CD1E01                             call conout ;another character printed
0252 D1C34402                           pop d! jmp print
                        ;
                        ;
                        read:
                                ;put prompt if in no echo mode
0256 3A9803B7C2                 lda echo! ora a! jnz read1
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
025D D5              push d
025E 11C303CDE6      lxi d,prompt! call msg          ;output prompt
0264 D10E0A          pop d! mvi c,creadf             ;set for read call
                read1:
                     ;read console buffer
0267 E1              pop h                           ;throw away return address
0268 D5               push d
0269 CD0900          call next                       ;make the call
026C E1234604        pop h! inx h! mov b,m! inr b     ;get the buffer length
0270 05CA8002  putnxt:               dcr b! jz read2
0274 235EC5E5                  inx h! mov e,m! push b! push h
0278 CD1E01E1C1                call conout! pop h! pop b       ;put character
027D C37002                    jmp putnxt

0280 3AE502B7F5read2: lda input! ora a! push psw
0285 1E0DC41E01      mvi e,cr! cnz conout                    ;call if putting input
028A F11E0AC41E      pop psw! mvi e,lf! cnz conout           ;call if putting input
0290 C30301          jmp retmon1


                ;
00E1 =          func1: equ     conin
                ;
011E =          func2: equ     conout
                     ;write console character
                ;
011E =          func5: equ     listf
                     ;write list character
                     ;write to list device
                ;
00CB =          func6: equ     rawio
                ;
0244 =          func9: equ     print
                     ;write line until $ encountered
                ;
0256 =          func10:        equ     read
                ;
0108 =          func11:        equ     retmon0
                ;
015F =          func13:        equ     close
                ;
015F =          func37:        equ     close
                ;
015F =          func98:        equ     close
                ;
                FUNC111:                       ;PRINT BLOCK TO CONSOLE
                FUNC112:                       ;LIST BLOCK
0293 EB5E235623      XCHG! MOV E,M! INX H! MOV D,M! INX H
0298 4E2346EB        MOV C,M! INX H! MOV B,M! XCHG
                     ;HL = ADDR OF STRING
                     ;BC = LENGTH OF STRING
                BLK$OUT:
029C 78B1C8          MOV A,B! ORA C! RZ      ;is length 0, return if so
029F C5E5            PUSH B! PUSH H
02A1 5ECD1E01        mov e,m! call conout    ;put character
02A5 E123C10B        POP H! INX H! POP B! DCX B
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
02A9 C39C02          JMP BLK$OUT

                ;       end of BDOS Console module

                ;*****************************************************************
                ;               DATA AREA
                ;*****************************************************************

02AC 00         exit$off        db      0       ;offset in exit$table of destination

                trap$tbl:
                        ;function dispatch table (must match func$tbl below)
                ;       db      lchrf, lblkf, coutf, cstatf, crawf
                ;       db      pbuff, cinf, creadf, resetf, resdvf
                ;       db      resalvf, pblkf, eot

02AD 1E01               dw      func5           ;function 5   - list output
02AF 9302               dw      func112         ;function 112 - list block
02B1 1E01               dw      func2           ;function 2   - console output
02B3 0801               dw      func11          ;function 11  - console status
02B5 CB00               dw      func6           ;function 6   - raw console I/O
02B7 4402               dw      func9           ;function 9   - print string
02B9 E100               dw      func1           ;function 1   - console input
02BB 5602               dw      func10          ;function 10  - read console buffer
02BD 5F01               dw      func13          ;function 13  - disk reset (close first)
02BF 5F01               dw      func37          ;function 37  - drive reset (close first)
02C1 5F01               dw      func98          ;function 98  - reset allocation vector
02C3 9302               dw      func111         ;function 111 - print block

                ;*****************************************************************
                ;       Following variables and entry points are used by PUT.COM
                ;       Their order and contents must not be changed without also
                ;       changing PUT.COM.
                ;*****************************************************************

                movstart:
                init$table:                     ;addresses used by PUT.COM for initial.
                scbadr:                                 ;address of System Control Block
02C5 0E00               dw      kill            ;kill flag for error on file make
                                                ;(passed to PUT.COM by RSX init function)
                        ;
                        if bios$functions       ;PUT.RSX initialization
                        ;
02C7 4B         gobios:         mov     c,e
02C8 C3                 db      jmp
02C9 2A00       biosout         dw      bios$output     ;set to real BIOS routine
                                                ;(passed to PUT.COM by RSXFUNC)
                biosjmp
02CB 0C0D               dw      warm$trap       ;address of bios jmp initialized by COM
                biosfix
02CD 0000               dw      0               ;address of jmp in resbdos to restore
                                                ;restore only if changed when removed.
02CF C3         wstart:         db      jmp
02D0 0000       wmsta:  dw      0               ;address of real warm start routine
02D2 0000       wmjmp:  dw      0               ;address of jmp in bios to restore
02D4 0000       wmfix:  dw      0               ;address of jmp in resbdos to restore
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        bdosfunc:
02D6 02                         db      coutf
                        restore$mode
02D7 00                         db      0               ;0FFh = no bios restore, 07fh = restore
                                                        ;only bios jmp, 0 = restore bios jump and
                                                        ;resbdos jmp when removed.
                                endif
                        ;
                        ;       equates function table
                        ;
00FF =                  eot     equ     0ffh    ; end of function table
00FE =                  skipf   equ     0feh    ; skip this function
                        ;
                        ;
                        func$tbl:               ;no trapping until initialized by PUT.COM
02D8 FF00000000                 db      eot,0,0,0,0,0,0,0,0,0,0,0,0
                        ;       db      lchrf, lblkf, coutf, cstatf, crawf
                        ;       db      pbuff, cinf, creadf, resetf, resdvf
                        ;       db      resalvf, pblkf, eot
                                ;
02E5 00                 input   db      0               ;put console input to a file
02E6 00                 list    db      0               ;intercept list functions
                        echoflg:
02E7 01                         db      1               ;echo output to device
                        cooked:                                 ;must be next after echo
02EB 00                         db      0               ;TRUE if ctrl chars (except ^Z) placed
                                                        ;in the output file
                        rsxfunctions:
02E9 FF                 killf:  db      0ffh            ;not used until PUT initialized
02EA FF                 fcbf:   db      0ffh            ;not used until PUT initialized
02EB 00                 record:         db      0               ;counts down records to block boundary
02EC 00                 blm:    db      0               ;block mask = records per block (rel 0)
                        program:                        ;this flag must be @ .PUTFCB-2
02ED 00                         db      0               ;true if put program output only
02EE 00                 putusr:         db      0               ;user number for redirection file
02EF FF                 putfcb:         db      0ffh            ;preset to 0ffh to indicate not active
02F0 5359534F55                 db      'SYSOUT  '
02F8 242424                     db      '$$$'
02FB 0000                       db      0,0
02FD 00                 putmod:         db      0
02FE                    putrc:  ds      1
02FF                            ds      16              ;map
030F                    putcr:  ds      1
                                ;
0310 00                 cbufp   db      0               ;current character position in cbuf
                        movend:
                        ;*****************************************************************

                        cbuf:                           ;128 byte buffer (could be ds 128)

0311 0303030303                 db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
0321 0303030303                 db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
0331 0303030303                 db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
0341 0303030303                 db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3

0351 0303030303                 db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 841  0361 0303030303        db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
 842  0371 0303030303        db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
 843  0381 0303030303        db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
 844
 845                         ;
 846                         if bios$functions
 847                         ;
 848                 exit$table:                     ;addresses to go to on exit
 849  0391 0900              dw      next            ;BDOS
 850  0393 C702              dw      gobios
 851                         endif
 852                         ;
 853  0395 8000      udma:   dw      buf             ;user dma
 854  0397 00        user:   db      0               ;user user number
 855  0398 00        echo:   db      0               ;echo output to console flag
 856  0399 01        ccpcnt:         db      1               ;start at 1 (decremented each CCP)
 857                 sav$area:                       ;14 byte save area
 858  039A 6868686868        db      68h,68h,68h,68h,68h, 68h,68h,68h,68h,68h
 859  03A4 68686868          db      68h,68h,68h,68h
 860                 close$err:
 861  03A8 0D0A505554        db      cr,lf,'PUT ERROR: FILE ERASED',cr,lf,'$'
 862  03C3 0D0A505554prompt:         db      cr,lf,'PUT>$'
 863                         ;
 864                 patch$area:
 865  03CA                   ds      30h
 866  03FA 2031353132        db      ' 151282 '
 867  0402 20434F5059        db      ' COPYR ''82 DRI '
 868
 869  0411 6767676767        db      67h,67h,67h,67h, 67h,67h,67h,67h, 67h,67h,67h,67h
 870  041D 6767676767        db      67h,67h,67h,67h, 67h,67h,67h,67h, 67h,67h,67h,67h
 871  0429 6767676767        db      67h,67h,67h,67h, 67h,67h,67h,67h
 872                         ;
 873                 stack:                          ;16 level stack
 874                 oldstack:
 875  0431 0000              dw      0
 876  0433                   end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ADDHLA          01F6  224  420  578#
BDOS            0005   46#  47  129
BDOSFUNC        02D6  166  785#
BDOSL           0006   47#  500  506
BDOSOFF         00FE  109#  509
BIOSFIX         02CD  547  778#
BIOSFIXUP       01BC  504  526#
BIOSFUNCTIONS   FFFF   39#  162  182  210  524  770  846
BIOSJMP         02CB  540  776#
BIOSOUT         02C9  538  774#
BIOSOUTPUT      002A  160#  774
BIOSTRAP        0035  169  185#
BLKOUT          029C  724#  729
BLM             02EC  439  816#
BUF             0080   48#  853
CBUF            0311  454  833#
CBUFP           0310  143  416  433  829#
CCPCNT          0399  288  856#
CCPFLG          00B3  107#  590
CCPRES          0020  108#  592
CINF            0001   69#
CKCCP           01FB  283  587#
CKECHO          00BA  298  306#
CKLIST          00A6  284  295#
CLOSE           015F  437  447#  488  613  712  714  716
CLOSEERR        03A8  489  860#
CLOSEF          0010   78#  448
CONIN           00E1  328  336#  694
CONIN1          00E4  332  338#
CONINFX         0074  117#
CONMODE         00CF  111#
CONOUT          011E  340  358  394#  666  684  688  689  696  727
CONOUTF         011E  395#
CONSTFX         006E  116#
COOKED          02E8  649  809#
COUTF           0002   70#  354  786
CR              000D   63#  345  652  688  861  861  862
CRAWF           0006   71#
CREADF          000A   72#  675
CSTATF          000B   73#
CTL             005E   65#  404
CTLC            0003   52#
CTLE            0005   53#
CTLH            0008   54#  655
CTLOUT          011E  396#
CTLOUT1         012A  398  402#
CTLP            0010   55#
CTLR            0012   56#
CTLS            0013   57#
CTLU            0015   58#
CTLX            0018   59#
CTLZ            001A   60#  146  398
DCNT            00E1  115#
DELF            0013   79#
DMAF            001A   82#  197  554
DMAFUNC         0204  198  602#
```

COPYRIGHT © 1932
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

DREADF 0014 80#
ECHO 0398 308 351 371 672 855#
ECHOC 022F 399 647#
ECHOFLG 02E7 306 807#
EOT 00FF 795# 800
ERRFLG 00AA 110#
EXIT 005B 193 208# 249 337 384 633
EXIT1 005E 212# 216#
EXITOFF 02AC 187 215 737#
EXITTABLE 0391 216 848#
FALSE 0000 37# 40
FCBF 02EA 625 814#
FLUSHF 0030 85#
FUNC1 00E1 694# 751
FUNC10 0256 708# 752
FUNC11 0108 710# 748
FUNC111 0293 718# 756
FUNC112 0293 719# 746
FUNC13 015F 712# 753
FUNC2 011E 696# 747
FUNC37 015F 714# 754
FUNC5 011E 699# 745
FUNC6 00CB 703# 749
FUNC9 0244 705# 750
FUNC98 015F 716# 755
FUNCTBL 02D8 239 799#
GFCBF 0082 94#
GINITF 0080 92#
GKILLF 0081 93#
GOBIOS 02C7 772# 850
GOFUNCT 00C0 312#
INITTABLE 02C5 621 765#
INPUT 02E5 356 392 687 805#
INTERCEPT 007D 244 265#
JFCBF 008E 102#
JINITF 008C 100#
JKILLF 008D 101#
KILL 000E 130# 496 767
KILLF 02E9 630 813#
LBLKF 0070 91#
LCHRF 0005 74#
LF 000A 64# 353 653 689 861 861 862
LIST 02E6 296 806#
LISTCP 00D4 113# 299
LISTF 011E 393# 699
LOADF 003B 87# 519
MOV7 01EB 461 463 475 477 564#
MOVE 01ED 568# 573
MOVEND 0311 830#
MOVSTART 02C5 764#
MSG 01E6 490 559# 674
NBANK 000F 131#
NEXT 0009 128# 212 329 355 471 505 521 555 560 606
680 849
OLDSTACK 0431 270 363 382 874#
OPENF 000F 77#

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

OUTDEL 00D3 112# 662
PATCH 001A 134#
PATCHAREA 03CA 864#
PBLKF 006F 90#
PBUFF 0009 75# 559
PCFCBF 0086 98#
PCKILLF 0085 96#
PINITF 0084 95# 620
PLFCBF 008A 99#
PLKILLF 0089 97#
PREV 000C 129#
PRINT 0244 659# 667 705
PROGRAM 02ED 281 817#
PROMPT 03C3 674 862#
PUTC 0134 147 400 404 413#
PUTCHR 0119 325 390#
PUTCR 030F 827#
PUTDOS 0161 430 452#
PUTEOF 001B 142# 148 290 632
PUTFCB 02EF 627 820#
PUTMOD 02FD 824#
PUTNXT 0270 682# 685
PUTRC 02FE 825#
PUTUSR 02EE 467 819#
RAWIO 00CB 320# 703
READ 0256 670# 708
READ1 0267 672 676#
READ2 0280 682 687#
RECORD 02EB 434 815#
REDIRECT 0087 273#
REMOVERSX 0000 40# 503
RESALVF 0062 89#
RESDVF 0025 84# 203
RESETF 000D 76#
RESTOR 019C 145 431 438 487#
RESTOREMODE 02D7 529 787#
RETMON 0109 313 369#
RETMONO 0108 366# 710
RETMON1 0103 331 346 361# 373 690
RETMON2 00EF 348#
RNAME 0010 132#
RSXF 003C 88# 195
RSXFUNC 0215 196 618#
RSXFUNCTIONS 02E9 492 812#
RUBOUT 007F 61#
SAVAREA 039A 459 857#
SCBA 003A 106#
SCBADR 02C5 277 457 508 589 661 766#
SCBF 0031 86#
SERIAL 0000 124#
SETDMA 01E1 455 480 554#
SETECHO 00BD 302 307#
SKIP 0110 287 291 305 378#
SKIPF 00FE 796#
SPACE 0018 133#
STACK 0431 271 612 873#

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

TAB	0009	62#	654		
TBLJMP	0061	219#	317		
TBLSRCH	006A	200	202	204	231#
TBLSRCH1	006F	241#	248		
TRAP	0034	127	180#		
TRAPJMP	0006	126#	191		
TRAPTBL	02AD	316	739#		
TRUE	FFFF	36#	39		
UDMA	0395	478	604	853#	
USER	0397	854#			
USERF	0020	83#			
USRCODE	00E0	114#	462	466	476
WARMTRAP	020C	611#	777		
WBOOT	0000	44#	45		
WBOOTA	0001	45#			
WMFIX	02D4	545	784#		
WMJMP	02D2	534	783#		
WMSTA	02D0	532	782#		
WRITE	0141	428#			
WRITEF	0015	81#	429		
WSTART	02CF	614	781#		

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____